华章数学译丛

HZ BOOKS 华章教育

48

PEARSON

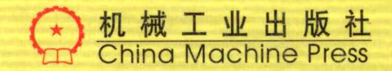
Modeling Derivatives Applications in Matlab, C++, and Excel

金融衍生品建模

基于Matlab、C++和Excel工具

(美) Justin London 著

郭梁 黄茜 等译
海通期货研究所



华 章 数 学 译 丛

48

Modeling Derivatives Applications in Matlab, C++, and Excel

金融衍生品建模

基于Matlab、C++和Excel工具

(美) Justin London

郭梁 黄茜 等译
海通期货研究所

机械工业出版社
China Machine Press

本书主要讲述重要的衍生品定价模型，并运用 Matlab、C++和 Excel 对包括信用衍生品（如信用违约掉期和信用关联记录）、债务抵押债券（CDO）、住房抵押贷款支持证券（MBS）、资产支持证券（ABS）、互换、固定收益证券，以及日渐重要的天气、电力、能源衍生品等进行建模. 本书提供了 Matlab 和 C++的示例代码，这些代码都可以更改和扩展，以满足实际需要. 读者将从衍生品模型的数据、理论和代码执行上获益.

本书适合作为统计、金融数学、经济管理等相关专业的教材，也可供对金融衍生品建模感兴趣的读者参考.

本书版权登记号：图字：01-2009-5638

图书在版编目（CIP）数据

金融衍生品建模：基于 Matlab、C++和 Excel 工具/(美) 伦敦（London, J.）著；郭梁等译. —北京：机械工业出版社，2010.9

(华章数学译丛)

书名原文：Modeling Derivatives Applications in Matlab, C++, and Excel

ISBN 978-7-111-31296-3

Ⅰ. 金… Ⅱ. ①伦… ②郭… Ⅲ. ①计算机辅助计算-软件包，Matlab-应用-金融市场-经济模型 ②C语言-程序设计-应用-金融市场-经济模型 ③电子表格系统，Excel-应用-金融市场-经济模型 Ⅳ. ①F830.9-39

中国版本图书馆 CIP 数据核字（2010）第 133215 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：王春华
北京市荣盛彩色印刷有限公司印刷
2011 年 1 月第 1 版第 1 次印刷
186mm×240mm・28.25 印张
标准书号：ISBN 978-7-111-31296-3
定价：65.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991；88361066
购书热线：(010) 68326294；88379649；68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

译 者 序

30 年前，金融衍生品还是一个深奥且专业的课题. 作为金融衍生品市场最发达的国家，美国创造出了很多金融衍生品工具，如金融期货、期权、货币互换等. 随着人们对金融衍生品的需求逐渐增加，世界各地还出现了若干新兴的金融市场，如芝加哥国际货币市场、新加坡国际货币交易中心、伦敦国际金融期货交易所、法国国际金融期货交易所等，人们在其中从事金融衍生品交易. 不可否认，金融工具在防范投资风险或借贷风险方面有明显的作用，不少新的工具突破了传统上对金融机构在业务范围、利率方面的管制. 各种金融期货、期权、货币互换等新的金融工具相继出现，提供了更为广泛的资金筹措渠道，同时降低了融资成本，这在一定程度上规避或减轻了投资风险. 金融衍生品市场的发展，促进了金融业的进一步发展，促进了经济与金融的国际化.

2008 年，一场突如其来的全球金融危机使得衍生品成为众人瞩目的焦点，有一些人认为作为衍生品的信用违约掉期（CDS）是导致金融危机的罪魁祸首之一. 尽管市场跌宕起伏，质疑声尚未停歇，但衍生品市场整体仍呈增长趋势. 根据美国期货业协会（FIA）对全球 69 个期货、期权交易所的数据统计，2008 年全球场内衍生品成交量达 176.5 亿张，同比增长 13.7%，其中期货成交量达 82.91 亿张，期权成交量达 93.61 亿张.

无论与欧美发达国家相比，还是与相同类型的发展中国家相比，我国在金融衍生品交易方面的落后是毋庸置疑的. 股指期货、利率期货、汇率期货以及相应的期权交易已经成为大多数经济体不可或缺的一部分. 国内利率和汇率尚未完全市场化，利率、汇率衍生品的推出仍遥遥无期. 股指期货准备了好几年，但何时推出仍然没有确切的时间表.

时至今日，对国内投资者而言，用数量工具构建衍生品模型仍然比较陌生. 金融衍生品的缺乏导致国内对衍生品的建模、定价等知之甚少，这方面的书籍也相当有限. 因此，当海通期货研究所决定翻译引进若干期货领域的权威著作并介绍给广大的中国读者时，我便自告奋勇地提出负责此书的翻译工作，希望能为推进中国衍生品市场的发展贡献自己的一份绵薄之力.

由于工作繁忙，以及尽快出版的需要，我和海通期货研究所的同事们一起完成了翻译任务. 本书翻译的分工是：前言、第 1 章由刘佳利负责翻译，第 2 章、第 3 章由王智泽负责翻译，第 4 章、第 5 章由龚劼负责翻译，第 6 章由黄茜、郭梁负责翻译，第 7 章由郭梁负责翻译，第 8 章、第 9 章由黄茜负责翻译，附录、参考文献由杨钤雯负责翻译. 译稿由郭梁、黄茜、简比佳审校.

郭梁

海通期货研究所

2010 年 8 月 7 日

前 言

随着新型的金融衍生品（如信用衍生品）的迅猛发展，上百家金融机构正在交易这些复杂的金融工具，并雇用了成千上万的金融、技术专业人员为他们建立有效而准确的模型. 由此带来的就是对于 C++和 Matlab（衍生品模型构建与执行所采用的两种常用语言）等编程工具的广泛应用，使从业人员拥有熟练应用这些程序语言进行编程的技能也变得愈加重要. 此外，Excel 作为当前许多金融机构自营部门的前端交易应用工具，熟练操作 Excel 也是非常重要的.

本书是第一本详细涵盖了重要的衍生品定价模型的书，书中运用 Matlab、C++和 Excel 对包括信用衍生品（如信用违约掉期和信用关联记录）、债务抵押债券（CDO）、住房抵押贷款支持证券（MBS）、资产支持证券（ABS）、互换（掉期）、固定收益证券，以及日渐重要的天气、电力、能源衍生品等进行建模. 读者将从衍生品模型的数据、理论和代码执行上获益.

本书列举了大量的 Matlab、C++和 Excel 的应用实例. 为了展示这些模型在实践中如何运用，书中所有例子皆来源于 Bloomberg 实时数据. 本书旨在教会读者怎样正确地开发和执行衍生品程序，以帮助他们在开发应用程序时能找到适合的程序代码. 最好的学习方法是参照例子运行代码. 各章内容分别如下：

- 第 1 章：互换和固定收益债券.
- 第 2 章：copula 和 copula 方法论.
- 第 3 章：住房抵押贷款支持证券.
- 第 4 章：债务抵押债券.
- 第 5 章：信用衍生品.
- 第 6 章：天气衍生品.
- 第 7 章：能源和电力衍生品.
- 第 8 章：运用 Matlab 执行能源衍生品模型. 本章内容是在休斯敦大学全球能源管理学院金融学教授、院长 Craig Pirrong 的原著基础上编写而成.
- 第 9 章：商业房地产支持证券（一种资产支持证券）. 在新加坡国立大学房地产金融系 Tien-Feo Sing 教授的原著基础上编写而成.

为了向读者提供多种不同视角和尽可能全面的信息，整本书汇集了多位行业骨干和专家的各类开发成果与模型. 本书不仅涵盖复杂的衍生品模型及所有编码，还结合了行业专家的重要成果. 例如，第 2 章和第 5 章提到了 Galiani(2003)，第 4 章提到了 Picone(2004)，第 1 章和第 3 章讨论了 Johnson(2004)，以及 Doerr(2002)、Xiang(2004) 和 Xu(2004) 的关于能源衍生品的宝贵研究. 在第 8 章，Craig Pirrong讨论了 PJ（Pirrong-Jermayakan）模型，一个双向交替的隐式有限差分法（ADI）对能源衍生品定价的求导方案. 第 9 章中，Tien-Feo Sing 利用蒙特卡罗模型给资产支持证券定价. 此外，致谢中提到的许多人都向本书提供了代码.

本书着重强调怎样利用 C++、Matlab 和 Excel 对价格、贸易和对冲交易这些复杂的模型进行执行和编码，并没有集中讨论设计模式或最好的编码技术（这些也许会在随后发行的版本

中讨论). 在构建面向对象的代码时，效率和模块化是重要的设计目标. 鉴于建立利息树时的一些常规化程序，本书一些例子中的 C++ 编码可能会更趋向于模块化. 本书的重点是提供适合读者的工作执行方式. 然而，本书也为如何建立有效的模型提供了一些论述和有用的技巧. 例如一个常见问题：当开发运用和存储多维数据模型时，如何进行数据结构的内存分配. 使用一个预定义的二维数组，由于它是固定大小的，所以不是最有效的分配内存的方法. 如果你不知道这种结构能存储多少实际数据，有很多内存也许会闲置和浪费. 另一方面，预定义数组的大小可能不够大.

虽然二维数组很容易定义，但使用数组模板类（能够处理多维）和 C++ 中标准模板库（STL）的向量更有效，因为它们是动态的，且不会占用过多内存. 本书运用了这些结构，当然也有一些二维数组. Matlab 作为一种矩阵操作语言，如果数组的大小没有预先定义，那么当数据被使用时，Matlab 将提供自动内存分配. 在 Matlab 里，所有的数据被当做矩阵对象，例如，单个的数字被当做一个 1×1 的阵列. 数据能从一个对象里加入或移走，这个对象将动态扩展或者减少内存空间.

尽管我们努力想把拼写错误都找出来，但鉴于本书的篇幅以及复杂性，不可避免地还是会有一些错误. 任何更正都会公示在网站上.

希望本书能带给你开发、构建、测试你自己模型的基础. 与此同时，通过利用前期测试代码，能节省大量的开发时间.

致谢

特别感谢以下人员提供的代码以及为此书所做的贡献：

Ahsan Amin

Sean Campbell

Francis Diebold

Uwe Doerr

Stefano Galiani

Michael Gibson

Stafford Johnson

Jochen Meyer

Dominic Picone

Craig Pirrong

Eduardo Schwartz

Tien Foo Sing

Liuren Wu

Lei Xiong

James Xu

目　　录

第1章　互换与固定收益工具

互换（或称为掉期）常被用于对冲资产负债表的利率风险敞口，以及债券或债务组合的利率风险敞口. 通过与资产负债表的固定收益资产与负债（如债券或债务）久期的一一匹配，互换可以使得资产负债表对利率波动风险免疫. 理想情况下，这种利用互换进行的对冲应该尽可能地一方面满足固定收益组合的久期匹配，另一方面还要满足固定收益组合的现金流匹配. 例如，假设某银行拥有一个浮动利率的贷款务组合，其久期为 5 年. 该银行可以签署一个久期同样为 5 年的互换合约（互换的久期为固定利率部分久期与浮动利率部分久期的差值），该互换合约的性质是以支出浮动利率的现金流来换取收入固定利率的现金流（通过该互换合约可以有效地将浮动利率的贷款转换为固定利率贷款，同时锁定固定的利息收入）. 通常来讲，由于没有完全一致的现金流匹配，这类交易总是存在一定的基差风险，例如，从贷款上收到的利息与互换上需要付出资金的并不完全一致. 但是，总体上讲，银行还是可以降低自身在收益率曲线平移方面所需要面对的风险. 此外，货币市场上的机构投资者也可以利用在芝加哥期货交易所（CBOT）上市的国债期货与互换期货来构造一个对冲工具，以对冲企业债、政府债组合的利率风险. 由于互换期货有较好的标准性与流动性，与直接用互换相比，互换期货正在成为一个对冲固定收益组合风险的更便宜、更有效的途径[1]. 本章主要探讨对冲利率风险的有关细节，以及债券组合如何利用互换与固定收益类工具（如期货等）.

1.1 节将介绍用欧洲美元期货来计算 LIBOR 互换利率. 1.2 节探讨短期国债与长期国债，包括它们的报价与定价方式. 1.3 节将讨论对收益率曲线的自举分析（bootstrap）来计算互换的贴现利率. 1.4 节将讨论利用固定收益工具来对冲债券头寸与利率变化. 1.5 节将讨论债券久期、修正久期，以及每基点美元价值（DV01）的计算，后者对计算互换的久期与修正久期是非常必要的，因为一个典型的固定换浮动利率互换包含两个部分，这两个部分分别相当于一个固定利率债券与一个浮动利率债券. 1.6 节将介绍利率期限结构. 1.7 节将探讨如何量化对收益率曲线的自举分析. 1.8 节将介绍如何利用 Matlab 进行自举分析，并给出例子，而在 1.9 节中，我们探讨在 Excel 环境中的自举分析. 1.10 节将介绍在 Matlab 环境中利用 BDT 利率模型和 HJM 利率模型进行互换的一般性定价. 1.11 节将介绍利用利率期限结构来给互换定价，比如从零息债券或付息债券现金流分析中取得的远期利率曲线. 在 1.12 节中，我们应用并定价固定换浮动互换，包括利用 C++计算久期与风险暴露. 最后，1.13 节则提供了一个在 Matlab 中定价百慕大互换的例子.

1.1　欧洲美元（利率）期货

有时用每年 3 月、6 月、9 月、12 月到期的欧洲美元期货[2] 来计算到期期限大于 1 年的互换的 LIBOR 互换零息利率. 欧洲美元的利率可以被用来计算较长期限的远期利率. 在美国，LIBOR 即期利率通常被用来定义到期期限大于 1 年的 LIBOR 零息利率曲线. 于是，欧洲美元期货典型地应用于到期期限在 1 年与 2 年之间的互换利率分析，有时也会应用于 5 年、7 年，甚至 10 年的到期期限. 此外，用来定义债券面值收益（par yield）的互换利率，常被用来计算到

期期限大于 1 年的零息利率曲线. 通过 LIBOR 即期利率、欧洲美元期货以及互换利率的组合，就可以利用自举分析得到 LIBOR 或者互换的零息曲线.

典型地，将欧洲美元期货的利率转换为远期利率需要做一个凸度调整. 对于较短的到期期限（小于 1 年）而言，欧洲美元期货利率可以被视同为相应的远期利率. 但是，对于更长的到期期限而言，期货与远期合约的差异在利率波动难以预测时就变得重要起来[3].

假设欧洲美元期货的报价为 P，那么一张合约的面值就是

$$10\,000(100-0.25(100-P)) \tag{1.1}$$

即相当于 10 000 乘以期货价格，其中期货价格为 100－0.25（100－P）. 假设 P=96.7，则合约价值为

$$10\,000(100-0.25(100-96.7))=\$991\,750$$

欧洲美元合约与短期国债合约类似，但也有一些重要的区别. 欧洲美元合约与短期国债合约的标的面值都是 1 000 000 美元. 然而，对于短期国债合约来讲，合约价值在到期日会趋近于 1 000 000 美元面值的 91 天到期的短期国债价值. 如果该合约被持有到期，标的证券会进行交割[4]. 欧洲美元期货合约则是在交割月的第三个星期三之前的第二个伦敦工作日进行现金清算[5]. 最后一次的对盘将合约的价值调整为

$$f_0=(\$1\,000\,000)\left(\frac{100-0.25R}{100}\right)=10\,000(100-0.25R)$$

这里 R 为当时 LIBOR 市场欧洲美元利率的报价[6]. 欧洲美元利率的报价等于欧洲美元以季度复利计算的 90 天实际利率[7]. 这不是一个贴现率. 正如赫尔指出的，“因此，欧洲美元期货合约是一个以利率为标的的合约，而短期国债期货合约则是一个以短期国债价格（或者说贴现率）为标的的合约.”[8]

1.2 短期国债与长期债券

短期国债是美国财政部发行的短期贴现债券. 该债券发行时，在面值的基础以一定的贴现折扣发行. 短期国债的报价为其面值 100 美元的折扣率. 到期时，持有者以短期国债的全部面值兑现. 累计利息的每个基点（或每日盘算）是以实际天数除以 360 来计算，这里是假设每年有 360 天，而利息以购买与卖出之间经过的实际天数来进行累计. 短期国债的报价是短期国债以 360 天计算的年化美元收益，以其面值的百分比来表示：

$$\frac{360}{n}(100-P) \tag{1.2}$$

这里 P 为一个面值为 100 美元还有 n 天到期的短期国债的美元价值[9]. 而贴现率并不等于短期国债的到期收益率. 如果 90 天短期国债的美元价值为 99，那么该短期国债的报价将是 1.00. 90 天的到期收益率将是 1/99，即 1.01%. 将之转换为：

$$\frac{1}{99}\times\frac{360}{90}=0.040\,4$$

或以实际天数除以一年 360 天为基点，每年 4.04%. 另外，还可以这样转换：

$$\frac{1}{99}\times\frac{365}{90}=0.040\,96$$

或以实际天数除以一年 365 天为基点，大约每年 4.10%. 两种利率的表达方式都是基于 90 天时间的季度复利计算法. 为了直接比较财政部利率与国债报价得出的收益率，通常每半年复利计算（即每 180 天复合计算）采用以实际天数除以 365 天为基点的方式. 复合利率就是所谓的债券同等收益率. 在这里，债券同等收益率就是

$$\frac{1}{99} \times \frac{365}{180} = 0.02048$$

对于短期国债（到期期限小于 182 天）来说，货币市场收益率可以用债券同等收益率乘以 360/365 来得到. 在这个例子里，它是 2.02%.

短期国债期货的交割或交易标的物为一个到期期限为 91 天、面值为 1 000 000 美元的短期国债. 它们被用来投机或者对冲短期利率风险. 短期国债期货的价格通过银行间货币市场（IMM）指数或贴现率 R_d 来报价：

$$\text{IMM} = 100 - R_d$$

理论上，短期国债定价要通过一个持有成本模型

$$f_0 = S_0^M (1 + R_f)^T \tag{1.3}$$

得到，这里

f_0 = 短期国债期货价格

T = 期货到期时间

S_0^M = 到期期限为 $M(= 91 + T)$ 的即期短期国债的价格

R_f = 无风险收益率或回购利率

根据 Johnson (2004)[10] 的观点，假设 161 天短期国债的即期利率是 5.7%，70 天的回购利率（或无风险收益率）是 6.38%，那么 70 天到期的短期国债期货合约的价格将是

$$f_0 = (97.5844)(1.0638)^{70/365} = 98.7487$$

这里

$$S_0^{161} = \frac{100}{(1.057)^{161/365}} = 97.5844$$

期货价格是受套利力量掌控的. 如果期货市场价格高于 f^*，套利者就会卖空期货合约买进即期短期国债. 举例来说，假设期货市场价格为 $f^{M=70/365} = 99$，某个套利者将会卖空期货，同意在 70 天之后以 99 元的价格卖出一个 91 天短期国债. 同时，他将会买进现货多头，以 6.38% 的利率借入 97.584 4 来购买报价在 97.584 4 的 161 天短期国债现货，借款期限为 70 天. 70 天后的到期日，这个套利者将会通过期货空头交割来以 $f^{M=70/365} = 99$ 的价格卖掉手中的短期国债（此时该短期国债距离到期日还有 91 天），并且把借入的钱还上，本息共计 $f^* = 98.74875$，赚取了现金（CF_T）2 513 美元：

$$\begin{aligned} CF_T &= f_0^M - f_0^* = f_0^M - S_0^M (1 + R_f)^T \\ &= 99 - 97.5844(1.0638)^{70/365} = 99 - 98.7487 = 0.2513 \end{aligned}$$

因此，现金流或利润是

$$CF_T = (\$1\,000\,000)\left(\frac{0.2513}{100}\right) = \$2\,513$$

注意，如果 $f^M = 99$，某货币市场经理若计划投资 70 天收益率为 6.38% 的短期国债，可以

通过购买 161 天的短期国债并卖空 70 天的短期国债期货来锁定卖出价格，这种方式能获得更大利润. 例如，还是利用之前的数据，如果一个货币市场经理本来计划投资 70 天 97.5844 金额的短期国债，他可以买入相应金额的 161 天短期国债，并以 99 的价格卖空相应的期货. 与原来 70 天收益率为 6.38%的短期国债相比，他的收益率将是 7.8%：

$$R=\left(\frac{99}{97.5844}\right)^{365/70}-1=0.078$$

如果市场价格低于 f^*，那么套利者将买入期货，卖空现货. 假设 $f^M=98$，套利者将买入期货多头，也就是同意在 70 天之后以 98 美元的价格买入到期期限为 91 天的短期国债；并且卖空现货，也就是借入 161 天的短期国债，以 97.5844 的价格卖出，将换得的现金以 6.38% 的收益率投资 70 天. 70 天之后的到期日，套利者将以 98（f^M）的价格买入短期国债期货（现在该国债期货的到期期限为 91 天），用这个买入的短期国债去平掉之前的现货空头，并且在整个投资过程中获得 98.74875（f^*），实现净收入现金流 7 487 美元.

$$CF_T=f_0^*-f_0^M=S_0^M(1+R_f)^T-f_0^M$$
$$=97.5844(1.0638)^{70/365}-98=98.7487-98=0.2513=0.7487$$

因此，现金流或利润是

$$CF_T=(\$1\,000\,000)\left(\frac{0.7487}{100}\right)=\$7\,487$$

如果持有成本模型成立，那么 70 天短期国债的即期利率（回购利率）将会等于 161 天短期国债多头与 70 天短期国债空头的合成利率（隐含回购利率）：

买 161 天短期国债 $S_0^{161}=97.5844$

短期国债的空头为 $f_0^M=f_0^*=98.74875$

$$R=\left(\frac{98.74875}{97.5844}\right)^{365/70}-1=0.0638$$

此外，如果持有成本模型成立，那么期货的到期收益率（YTM）将会等于隐含的远期利率 F. 后者锁定未来 70 天进行的 91 天的投资，如下所述：

1. 卖出 70 天短期国债，价格为 $S_0^{70}=98.821$.
2. 买入 $n=\frac{S_0^{70}}{S_0^{161}}=\frac{98.821}{97.5844}=1.01267$ 的 161 天短期国债，价格为 97.5844.
3. 70 天后，以 100 的价格平仓短期国债空头.
4. 再过 90 天后，平仓原来的 161 天短期国债，收到：1.01267（100）=101.267.

$$R=\left(\frac{101.267}{100}\right)^{365/91}-1=0.0518=F_{91,70}$$

相当于

$$YTM_f=\left(\frac{100}{98.74875}\right)^{365/91}-1=0.0518$$

1.2.1 利用短期国债期货避险

货币经理常用短期国债进行避险. 假设某货币经理预计将于 6 月收到一笔 500 万美元的现金流，而他计划用这笔资金投资 91 天的短期国债. 当时 6 月期的短期国债期货的 IMM 报在 91

(6 月 IMM=91，或 R_D=9%)，该货币经理可以通过买入 5.115 张 6 月期短期国债期货多头的方式锁定 9.56%的收益率：

$$f_0^{\text{June}} = (\$1\,000\,000)\left(\frac{100-(9)(0.25)}{100}\right) = \$977\,500$$

$$YTM_f = \left(\frac{\$1\,000\,000}{\$977\,500}\right)^{365/91} - 1 = 0.095\,6$$

$$n_f = \frac{CF_T}{f_0} = \frac{\$5\,000\,000}{\$977\,500} = 5.115 \text{ 张长期合约}$$

假设到了 6 月，91 天的短期国债即期利率为 8%. 该货币经理将发现短期国债的价格高于 980 995 美元，但是，通过期货的平仓则可以实现 17 877 美元的利润. 将这部分利润与 500 万美元的现金加在一起，该经理仍然可以买入 5.115 张短期国债[11]，并获得针对 500 万美元本金而言的 9.56%的收益率：

在 6 月合约到期时，短期国债即期利率为 8%

$$\text{即期利率} = S_T^{91} = \frac{\$1\,000\,000}{(1.08)^{91/365}} = \$980\,995$$

$$\text{利润} = \pi_f = [\$980\,995 - \$997\,500](5.115) = \$17\,877$$

$$\text{对冲比率} = n_{TB} = \frac{CF+\pi_f}{S_T^{91}} = \frac{(\$5\,000\,000 + \$17\,877)}{\$980\,995} = 5.115$$

$$\text{合约价值} = f_0^{\text{June}} = (\$1\,000\,000)\left(\frac{100-(9)(0.25)}{100}\right) = \$977\,500$$

$$\text{收益率} = R = \left[\frac{(\$1\,000\,000)(5.115)}{\$5\,000\,000}\right]^{365/91} - 1 = 0.956 = 9.56\%$$

又设到了 6 月，91 天的短期国债即期利率为 10%. 该货币经理将发现短期国债的价格低于 976 518 美元，但是，通过期货的平仓会实现 5 025 美元的亏损. 将 500 万美元的现金支付完 5 025美元的亏损后，因为短期国债的价格更低了，该经理仍然可以买入 5.115 张短期国债，并获得针对 500 万美元本金而言的 9.56%的收益率：

在 6 月合约到期时，短期国债即期利率为 10%

$$S_T^{91} = \frac{\$1\,000\,000}{(1.10)^{91/365}} = \$976\,518$$

$$\pi_f = [\$976\,518 - \$997\,500](5.115) = -\$5\,025$$

$$n_{TB} = \frac{CF+\pi_f}{S_T^{91}} = \frac{(\$5\,000\,000 - \$5\,025)}{\$976\,518} = 5.115$$

$$f_0^{\text{June}} = (\$1\,000\,000)\left(\frac{100-(9)(0.25)}{100}\right) = \$977\,500$$

$$R = \left[\frac{(\$1\,000\,000)(5.115)}{\$5\,000\,000}\right]^{365/91} - 1 = 0.956 = 9.56\%$$

可以发现，无论利率如何变化，货币经理都可以获得 9.56%的收益率.

1.2.2 期货多头避险：对 182 天短期国债进行合成期货避险

假设某货币经理预计将于 6 月收到一笔 500 万美元的现金流，而他计划用这笔资金投资

182 天的短期国债. 由于短期国债期货标的合约的到期期限为 91 天，该经理需要同时买入 6 月与 9 月的短期国债期货多头（注意两个合约之间的间隔大约为 91 天）来锁定 182 天的短期国债收益. 如果当时 6 月期的短期国债期货的 IMM 报在 91，9 月期的短期国债期货的 IMM 报在 91.4，该经理将通过同时买入 5.115 张 6 月短期国债期货多头与 5.11 张 9 月短期国债期货多头的方式来锁定 182 天短期国债 9.3%的收益率：

$$6\text{ 月 IMM} = 91,\text{或 } R_D = 9\%$$

$$9\text{ 月 IMM} = 91.4,\text{或 } R_D = 8.6\%$$

$$f_0^{\text{June}} = (\$1\,000\,000)\left(\frac{100-(9)(0.25)}{100}\right) = \$977\,500$$

$$f_0^{\text{Sept}} = (\$1\,000\,000)\left(\frac{100-(8.6)(0.25)}{100}\right) = \$978\,500$$

$$YTM_f^{\text{June}} = \left[\frac{\$1\,000\,000}{\$977\,500}\right]^{365/91} - 1 = 0.095\,6$$

$$YTM_f^{\text{Sept}} = \left[\frac{\$1\,000\,000}{\$978\,500}\right]^{365/91} - 1 = 0.091$$

$$n_f^{\text{June}} = \frac{CF_T}{f_0} = \frac{\$5\,000\,000}{\$977\,500} = 5.115$$

$$n_f^{\text{Sept}} = \frac{CF_T}{f_0} = \frac{\$5\,000\,000}{\$978\,500} = 5.112$$

因此，收益是

$$YTM_f^{182} = [(1.095\,6)^{91/35}(1.091)^{91/365}]^{365/182} - 1 = 0.093$$

假设在 6 月，91 天短期国债利率为 8%，182 天短期国债利率为 8.25%. 根据这些利率，91 天现货短期国债的价格将是

$$S_T^{91} = \frac{\$1\,000\,000}{(1.08)^{91/365}} = \$980\,995$$

182 天现货短期国债的价格将是

$$S_T^{182} = \frac{\$1\,000\,000}{(1.08)^{182/365}} = \$961\,245$$

如果持有成本模型成立，那么 9 月份期货合约在 6 月份时的价格为

$$f_0^{\text{Sept}} = S_0^{182}(1+R_f)^T = \$961\,245(1.08)^{91/365} = \$979\,865$$

以这些价格，经理可以从期货上获取的利润为

$$6\text{ 月 } \pi_f = [\$980\,995 - \$977\,500]5.115 = \$17\,877$$

$$9\text{ 月 } \pi_f = [\$979\,865 - \$978\,500]5.11 = \$6\,975$$

通过同时平仓两种期货合约（后者抵消了短期国债期货价格的上升），获得了总计为 24 852 美元的利润，并可以购买

$$n_{TB} = \frac{\$5\,000\,000 + \$24\,852}{\$961\,245} = 5.227$$

182 天短期国债，收益率为

$$R = \left[\frac{(5.227)(\$1\,000\,000)}{\$5\,000\,000}\right]^{365/182} - 1 = 0.093$$

或 500 万美元基础之上的 9.3%. 读者可以验证，如果 91 天与 182 天短期国债利率上升，收益率仍将是 9.3%. 此外，国债期货合约需要交割或购买一张面值为 100 000 美元的国债，而期货合约允许交割的国债种类有许多. 因此，需要通过一个转换因子来确定被交割的国债所对应的期货价格. 在实践中，通常是交割时计算下来最便宜的国债被用来交割. 国债期货的报价是以一张票息 8%、每半年付息、15 年到期、面值为 100 美元的国债为基础的.

在 Matlab 中，我们可以通过在美国中长期国债表中将美国短期国债市场参数重设为零息债券的方式来直接比较短期国债与中长期国债，具体可通过以下函数来实现：

```
[TBondMatrix, Settle] = tbl2bond(TBillMatrix)
```

TBillMatrix 表示的是短期国债参数. 一个 $N \times 5$ 的矩阵，其中每一行表示一个短期国债，N 是短期国债的数量；每一列则分别是 [Maturity Days Maturity Bid Asked AskYield]，如表 1.1 所示.

表 1.1

参数	说明
Maturity	作为一连串的日期数字，使用 datenum 可将字符串转换为日期数字
DaysMaturity	到期日作为一个整数. 到期日以越日为基础表示，从结算日到期限日的真正天数是 DaysMaturity+1
Bid	给出银行贴现率. 贴现率按票据的买入面值报出，并且年利率以单利计算（小数位采取十进制方式）
Asked	给出银行贴现率（小数位采取十进制方式）
AskYield	卖方收益率：持有票据至到期日获得债券等值收益，并且年利率以单利计算，假定为 365 天（小数位采取十进制方式）

长期国债参数则由 **TBondMatrix** 来表示，一个 $N \times 5$ 的矩阵，其中每一行表示一个同等的长期国债（零息）；每一列则分别是 [CouponRate Maturity Bid Asked AskYield]，如表 1.2 所示.

表 1.2

参数	说明
CouponRate	票面利率总是维持在 0
Maturity	期限作为一连串的日期数字，与短期国债的到期日期相同
Bid	买方报价基于 $100 面值
Asked	卖方报价基于 $100 面值
AskYield	卖方收益率：有效回报从持有时间到期为止，年利率以复利计算

例 1 1997 年 12 月 22 日公布的短期国债市场参数：

```
TBill = [datenum('jan 02 1998') 10 0.0526 0.0522 0.0530
         datenum('feb 05 1998') 44 0.0537 0.0533 0.0544
         datenum('mar 05 1998') 72 0.0529 0.0527 0.0540];
```

执行如下函数：

```
TBond = tbl2bond(TBill)

TBond =
          0 729760 99.854 99.855 0.053
          0 729790 99.344 99.349 0.0544
          0 729820 98.942 98.946 0.054
```

1.3 在 Matlab 中计算短期国债价格与收益率

在 Matlab 中，你可以将短期国债的收益率指定为货币市场收益率或债券同等收益率.

Matlab 中短期国债函数假定每个短期国债的面值都为 100 美元. 短期国债的价格可用如下函数来计算：

```
Price = prtbill(Settle, Maturity, Face, Discount)
```

这里的讨论在表 1.3 中列出.

表 1.3

参 数	说 明
Settle	键入日期号码或字符串，Settle 必须在到期日当天或之前结算
Maturity	键入日期号码或字符串
Face	赎回价值
Discount	短期国债贴现率（小数位采取十进制方式）

例 2 短期国债价格计算如下：

```
Settle = '2/10/2005';
Maturity = '8/7/2005';
Face = 1000;
Discount = 0.0379;
Price = prtbill(Settle, Maturity, Face, Discount);

Price =
      981.2606
```

短期国债的到期收益率可以用 yldtbill 函数进行计算：

```
Yield = yldtbill(Settle, Maturity, Face, Price)
```

这个示例中的短期国债收益率为：

```
Yield = 0.0386
```

短期国债的债券同等收益率为：

```
BEYield = beytbill(Settle, Maturity, Discount)
```

这里 Discount 是指短期国债的贴现率，可以通过 discrate 函数来计算：

```
Discount = discrate(Settle, Maturity, Face, Price, Basis)
```

在这个例子里，Basis=2（实际天数/360 天的换算方式），所以

```
Discount = 0.0379
```

如预期的，债券同等收益率如下所示：

```
BEYield = 0.0392
```

1.4 对债券头寸进行套期保值

1.4.1 利用短期国债看涨期权对 91 天短期国债期货进行套期保值

根据 Johnson（2004）观点，假设一个投资者预期 6 月份短期利率上涨，同时也关注下跌的概率. 为了在高利率下取得投资收益，并进行套期保值以防止利率下跌，投资者可以买入现货短期国债的看涨期权，或者买入 6 月份短期国债期货的期权. 下面举例说明，假设 6 月份短期国债期货期权的执行价的报价为 90（是指 $X=975\ 000$ 时行权），报价 1.25（看涨期权的价格为 3 125 美元），6 月份到期的时候同时会发生 5 000 000 美元现金流入. 用期权对这个 91 天的头寸进行套期保值，投资者需要以 16 025.64 美元的价格买入 5.128 205 张看涨期权（假设可以除尽），具体的计算过程如下：

$$n_c = \frac{CF_T}{X} = \frac{\$5\ 000\ 000}{\$975\ 000} = 5.128\ 205 \text{ 张合约}$$

$$Cost = (5.128\ 205)(\$3\ 125) = \$16\ 025.64$$

$$\pi_c = 5.128\ 208\ 5[\text{Max}(S_T - \$975\ 000\ 0) - \$3\ 125]$$

如果在 6 月到期时，短期国债利率下跌，则投资者可以从期权中获利. 就像表 1.4 中所示，如果短期国债的贴现率小于等于 10%，投资者可以花费 500 万美元买入 5.112 张 91 天短期国债债券，并通过期权获利，锁定了面值为 500 万美元、到期收益率为 9.3% 的收益. 相反，如果短期国债利率上涨，投资者将受益于利率的上涨，损失的仅仅是看涨期权的价值（16 025.64美元）. 在这个例子中，如果贴现率大于 10%，当利率上升时，投资者可以购买更多的短期国债，从而得益于利率的上升. 因此，对于看涨期权的价格，投资者可以锁定面值为 500 万美元、到期收益率为 9.3% 的损失，换来因利率上升而带来的收益.

表 1.4 用 6 月短期国债期货的看涨期权来对冲 6 月 500 万美元的现金流

看涨 $X=90$（975 000），$C=1.25$（\$3 125），$n=5.128\ 205\ 1$

1	2	3	4	5
即期利率	现货价格	收益/损失	nTB	到期收益
8	980 000	9 615.384 56	5.112	0.093
8.5	978 750	3 205.128 19	5.112	0.093
9	977 500	−3 205.128 2	5.112	0.093
9.5	976 250	−9 615.384 6	5.112	0.093
9.75	975 625	−12 820.513	5.112	0.093
10	975 000	−16 025.641	5.112	0.093
10.25	974 375	−16 025.641	5.115	0.096
10.5	973 750	−16 025.641	5.118	0.098
10.75	973 125	−16 025.641	5.122	0.101
11	972 500	−16 025.641	5.125	0.104
11.25	971 875	−16 025.641	5.128	0.107

资料来源：Johnson S(2004)．

需要注意的是，如果对 182 天短期国债进行套期保值，原理是一样的，只不过需要买入 6 月和 9 月的看涨期权. 在 6 月到期日时，投资经理需要了解两个头寸，同时买入面值 500 万美元短期国债加上（减去）182 天短期国债期权的收益（损失）.

1.4.2 空头套期保值：管理到期日缺口

企业、市政、金融机构、经纪商和承销商计划将来某个时点卖出债券或者融资，并且锁定某一个特定的利率，这个时候就需要进行空头套期保值.空头套期保值的原理与前面的例子正好相反，货币市场资金管理不是买入短期国债，而是卖出所持有的 6 月短期国债（持有的短期国债可以是 91 天或者 182 天).为了锁定收益，投资者将卖空 6 月短期国债期货（如果计划卖出 91 天的短期国债）或者 6 月和 9 月期货（如果计划卖出 182 天的短期国债).如果短期利率上涨（下跌)，将会导致短期国债价格下跌（上涨)，投资者卖出短期国债将会得到较少的收益，但是当短期国债期货到期时，投资者会得到收益（损失).

空头套期保值的另外一个重要用途是锁定未来债务的利率.举例说明，一个小型银行存在到期日缺口问题，其放出的短期贷款平均到期日长于 CDs（用来融资的).假设 6 月银行发放贷款 100 万美元，180 天后到期.假设银行的客户喜欢投资 90 天的 CDs，而不是 180 天的，因此银行卖出了 100 万美元 90 天到期的 CDs，LIBOR 为 8.258%，90 天后（9 月）银行将拥有现金 $1 019 758=($1 000 000) $(1.082\ 58)^{90/365}$，为了筹措融资，银行将不得不卖出于 90 天后（9 月份）到期的价值 1 019 758 美元的 CDs，利率为当时的 LIBOR.如果没有套期保值，银行将暴露于利率市场风险下.如果短期利率上升，银行需要对 9 月 CDs 支付更多的利息，减少利差（从 100 万美元 180 天的贷款的利率减去为 CDs 支付的利息）收益；如果利率下降，银行将增加利差.

假设银行担心 9 月利率会上升，决定对 9 月卖出的 1 019 758 美元的 CDs 用欧元期货合约（IMM=92.1）进行套期保值.为了完全规避风险，银行需要卖空 1.039 51 张 9 月欧元期货合约，（假设完全整除）计算过程如下：

$$f_0^{\text{Sept}} = \frac{100-(7.9)(0.25)}{100}(\$1\,000\,000) = \$981\,000$$

$$n_f = \frac{\$1\,019\,758}{\$981\,000} = 1.039\,51 \text{ 张短期欧元合约}$$

当期货价格为 981 000 美元时，银行可以锁定 9 月 CDs 的利率为 8.1%：

$$YTM_f^{\text{Sept}} = \left(\frac{\$1\,000\,000}{\$981\,000}\right)^{365/91} - 1 = 0.081$$

$$YTM_{182} = [(1.082\,5)^{90/365}(1.081)^{90/36}]^{365/180} - 1 = 0.081\,7$$

相比第一次卖出的 CDs（6 月）的利率 8.25%，银行仅需要对 180 天的 CDs 支付 8.17% 的利率.当第一个 CDs 在 9 月到期时，银行将卖出一个新的 90 天 CDs（金额为 1 019 758 美元）来支付第一个信用违约（CD）债务.如果 9 月的 LIBOR 上升，银行需要对新的 CDs 支付更高的利息，但是将来会产生收益；同样，如果 9 月 LIBOR 下降，银行将对新发行的 CDs 支付较低的利息，但是将来会带来损失.当 9 月期的 CDs 在 12 月到期时，利率对需要融得的这部分资金产生的影响将与支付后者的利率恰好抵消，使银行获得一个固定数量的债务.如表 1.5 所示，在 9 月 LIBOR 利率为 7.5%或 8.7%的情景（任何利率下均成立）下，银行 12 月负债（最初的 180 天时间段之后）显示为 1 039 509 美元.

表 1.5

9 月 LIBOR (R)	0.075	0.085
(2) $S_T^{CD} = f_T^{Sept} = \$1M/(1+R)^{90/365}$	982 236 美元	979 640 美元
(3) $\pi_t = [981\,000 - f_T]1.039\,1$	−1 378 美元	1 413 美元
(4) 6 月的信用违约债务	1 019 758 美元	1 019 758 美元
(5) 之后 90 天所有的财务资金：第 (4) 项减去第 (3) 项	1 021 136 美元	1 018 345 美元
(6) 之后 90 天末尾的债务：第 (5) 项乘以 $(1+R)^{90/365}$	1 039 509 美元	1 039 509 美元
(7) 180 天期间的利率 $R_{CD}^{180} = [\$1\,039\,509/\$1\,000\,000]^{365/180} - 1$	8.17%	8.17%

资料来源：*Johnson R S.*

注意，180 天之后的债务为 1 039 509 美元，也就相当于 9 月的 90 天利率为 8.1%，180 天利率为 8.17%：

$$YTM_f^{Sept} = \left[\frac{\$1\,039\,509}{\$1\,019\,758}\right]^{365/90} - 1 = 0.081$$

$$R_{CD180} = \left[\frac{\$1\,039\,509}{\$1\,000\,000}\right]^{365/180} - 1 = 0.081\,7$$

1.4.3 到期日缺口和持有成本模型

在前面的例子中，银行出现了到期日缺口，是由于银行发放的是 180 天 CDs 贷款，而银行的投资者更喜欢投资 90 天的 CDs，这就导致融资得到的资金与发放的贷款存在到期日缺口. 如果银行不存在到期日缺口，那问题就不存在了，银行可以通过发行 180 天的 CDs 融资用来发放 180 天的贷款，或者发行 90 天的 CDs 融资用来发放 90 天的贷款. 然而，假设 9 月的欧元期货价格超过它的持有成本，在这种情况下，银行发现为了发放 180 天的贷款，融资方式有两种，一种是卖出 180 天的 CDs，另外一种方式是发行合成 CDs，即卖出 6 月的 90 天 CDs，在到期时再滚动发行 9 月 90 天的 CDs，第二种方式融资更为便宜. 举例说明，如果 6 月 180 天 CDs 的即期利率交易价格为 96，利率为 8.63%，则 9 月欧元合约的持有成本为 97.897 ($97.897=96\,(1.082\,58)^{90/365}$). 如果 9 月期货合约价格为 98.1，则银行使用合成 CDs 的成本更低. 同样，如果期货价格低于持有成本，则 180 天的合成 CDs 价格将超过 180 天的 CDs，银行使用 180 天的 CDs 成本更低. 最后，如果欧元期货价格与持有成本的价格相同，则银行使用 CDs 和合成 CDs 的成本是一样的.

1.4.4 使用欧元看跌期权管理到期日缺口

如果不是用欧元期货对未来 CDs 进行套期保值，银行可以选择买入欧元、短期国债，或者买入欧元期货、短期国债期货的看跌期权. 在前面的例子中，假设银行对 9 月 CDs 销售进行套期保值，可以买入 9 月短期国债期货的看跌期权，到期时间与 6 月 CDs 是一致的，执行价格为 90 ($X=\$975\,000$)，期权价格为 0.5 ($C=\$1\,250$). 为抵消 9 月到期的 1 019 758 美元 6 月 CDs 的债务，银行需要买入 1.046 张 9 月短期国债期货看跌期权，成本为 1 307 美元，以套期保值 9 月 CDs.

$$n_p = \frac{CF_T}{X} = \frac{\$1\,019\,758}{\$975\,000} = 1.045\,905\,6 \text{ 张看跌期权}$$

$$Cost = (1.046)(\$1\,250) = \$1\,307$$

如果 9 月到期时利率较高，以至于短期国债的贴现率高于 10%，那么银行将从看跌期权中获益. 这样会减少融资成本. 如表 1.6 所示，如果短期国债的贴现率大于等于 10%，银行 90 天 CDs 利率比短期国债高 0.25%，则银行可以锁定 90 天后的债务 1 047 500 美元，180 天的有效利率为 9.876%.

另外，如果利率下降，使得现货短期国债的贴现率小于 10%，银行可以用更低的利率来融资1 047 500美元，而购买短期国债期货看跌期权的损失也只有 1 307 美元. 因此，使用期权进行套期保值，银行可以锁定未来最大的支付利率.

表 1.6

短期国债套期保值到期日缺口

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
R_D	S_Y	$Rate_{CD}$	π_P	信用违约 (CD) 债务	套期保值 (5)-(4)	90 天后的债务 $(6)[1+(3)]^{90/345}$	利率 $[(7)/1M]^{365/180}$
7%	\$982 500	0.075 88	−1 307	1 019 750	1 021 065	1 039 646	8.203%
8%	\$980 000	0.086 90	−1 307	1 019 750	1 021 065	1 042 262	8.756%
9%	\$977 500	0.098 07	−1 307	1 019 750	1 021 065	1 044 893	9.313%
10%	\$975 000	0.109 40	−1 307	1 019 750	1 021 065	1 047 500	9.867%
11%	\$972 500	0.120 80	1 307	1 019 750	1 018 451	1 047 500	9.867%
12%	\$970 000	0.132 40	3 922	1 019 750	1 015 836	1 047 500	9.867%

假定 90 天信用违约利率为 25%，这比短期国债利率还高：

$$Rate_{CD}=\left[\frac{\$1M}{S_T}\right]^{365/91}+0.0025-1$$

$$\pi_P=1.4059056[\mathrm{Max}[\$975000-S_r,\ 0]-\$1307]$$

资料来源：Johnson R S(2004) .

1.4.5 空头套期保值：对变动利率贷款进行套期保值

假如一家公司从银行获得 100 万美元的变动利率贷款，期限为 1 年，根据贷款协议，贷款从 9/20 开始利率为 9.5%，在 12/20、3/20 和 6/20 利率重置，重置利率为 LIBOR（年）加上 150 个基点（1.5%）除以 4，即（LIBOR+0.015)/4.

对于银行，这笔贷款是一个变动利率资产，同时，可以通过发行与 LIBOR 挂钩的 90 天 CDs 对这个变动利率进行套期保值. 对于公司，他们要承担利率风险（除非他们将贷款用于变动利率资产融资). 为了对变动利率进行套期保值，公司可以卖出一系列的欧元期货合约. 对于这个例子，假设公司分别在 12/20、3/20 和 6/20 卖空期货合约，具体价格如表 1.7 所示.

表 1.7

T	12/20	3/20	6/20
IMM 指数	91.5	91.75	92
f_0（每 \$100 面值）	97.875	97.937 5	98

用欧元期货合约锁定的利率是 100 减去 IMM 指数加上贷款基点：

$$锁定利率 = [100 - \text{IMM}] + [BP/100]$$
$$12/20: R_{12/20} = [100 - 91.5] + 1.5\% = 10\%$$
$$3/20: R_{3/20} = [100 - 91.75] + 1.5\% = 9.75\%$$
$$6/20: R_{6/20} = [100 - 92] + 1.5\% = 9.5\%$$

例如，假定 12/20，LIBOR 为 9%，IMM 指数价格为 91，期货收盘价为 97.75（面值 100 美元），在当日，公司将从看空的期货合约获利 1 250 美元. 即

$$f_0 = \frac{(100-(100-91.5))(0.25)}{100}(\$1\,000\,000) = \$978\,750$$

$$f_T = \frac{(100-(100-91))(0.25)}{100}(\$1\,000\,000) = \$977\,500$$

$$12/20\ 合约获利 = \$978\,750 - \$977\,500 = \$1\,250$$

在 12/20，公司为下一个季度支付的利息是 26 250 美元：

$$12/20\ 利息 = \left[\frac{(\text{LIBOR}+0.015)}{4}\right](\$1\,000\,000)$$
$$= \left[\frac{(0.09+0.015)}{4}\right](\$1\,000\,000) = \$26\,250$$

从 26 250 美元中减去 1 250 的收益（忽略时间价值因素），公司套期保值为下一季度支付的利息为25 000美元，年化利率后为 10%：

$$套期保值利率 = R^A = \frac{4(\$25\,000)}{\$1\,000\,000} = 0.10$$

另外，如果 12/20 LIBOR 为 8%，支付的季度利息为 23 750 美元（(0.08+0.015)/4)(1 000 000)=23 750). 这个收益是在公司的收益中减去期货合约 1 250 美元的损失（8%，$f_T=$ 98 000，12/20 合约的收益为 978 750−980 000=−1 250). 因此，对于公司而言，每个月的债务为 25 000 美元（23 750+1 250). 忽略资金的时间价值，公司年化的套期保值利率为 10%. 公司的 12/20 欧元空头期货合约价格为 91.5，锁定每季度 25 000 美元债务的年化利率为 10%. 如果 12/20LIBOR 为 9%，公司下一个季度需要支付 26 250 美元，但是公司在 12/20 从欧元期货中获利 1 250 美元，可以抵消一部分费用，使得有效的套期保值利率为 10%. 支付的利息、期货收益和实际利息计算如下：

$$12/20: \text{LIBOR} = 9\%$$

期货交易：

$$结算价格: S_T = 100 - \text{LIBOR}$$
$$S_T = 100 - 9(0.25) = 97.75$$
$$\pi_f = \frac{97.875 - 97.75}{100}(\$1\,000\,000) = \$1\,250$$
$$利息 = \frac{\text{LIBOR} + 150BP}{4}(\$1\,000\,000)$$
$$= \frac{0.09+0.015}{4}(\$1\,000\,000) = \$26\,250$$
$$实际利息 = \$26\,250 - \$1\,250 = \$25\,000$$

$$实际利率 = R^A = \frac{4(\$25\,000)}{\$1\,000\,000} = 0.10$$

如果 12/20LIBOR 为 6%，公司下一个季度仅仅支付 18 750 美元，12/20 欧元期货合约将损失 6 250 美元. 支付的利息和期货合约的损失使得有效的套期保值利率为 10%：

$$12/20\text{:LIBOR} = 6\%$$

期货交易：

$$结算价格: S_T = 100 - \text{LIBOR}$$

$$S_T = 100 - 6(0.25) = 98.5$$

$$\pi_f = \frac{97.875 - 98.5}{100}(\$1\,000\,000) = -\$6\,250$$

$$利息 = \frac{\text{LIBOR} + 150BP}{4}(\$1\,000\,000)$$

$$= \frac{0.06 + 0.015}{4}(\$1\,000\,000) = \$18\,750$$

$$实际利息 = \$18\,750 + \$6\,250 = \$25\,000$$

$$实际利率 = R^A = \frac{4(\$25\,000)}{\$1\,000\,000} = 0.10$$

公司用欧元期货合约对变动利率贷款进行套期保值，一年的固定利率是 9.687 3%：

$$贷款利率 = [(1.095)^{0.25}(1.10)^{0.25}(1.0975)^{0.25}(1.095)^{0.25}]^1 - 1 = 0.096\,873$$

需要注意的是，实际上，公司可能已经得到一年的固定利率贷款. 如果有了期货合约，公司可以有两个选择，一种是固定利率贷款，另外一种是合成固定利率贷款（变动利率贷款加上欧元期货合约看跌期权），后者成本更低. 公司可以使用一系列的欧元看跌期权和期货看跌期权对变动利率贷款进行套期保值. 使用看跌期权套期保值，每个季度公司锁定的最大支付利率随着实际利率的下降而下降.

1.5 债券与互换久期、修正久期，以及每基点美元价值 (DV01)

久期是以平均时间（年）发生的现金流，按照目前的收益率折现成现值，再用每笔现值乘以其距离债券到期日的年限求和，然后以这个总和除以债券目前的价格得到的数值. 如果 C_i 是在 t_i 时间总计支付的现金流，久期的计算公式为：

$$久期 = \frac{\sum_{i=1}^{n} \frac{t_i C_i}{(1 + y_i)^{t_i}}}{B} \tag{1.4}$$

这里 n = 现金流数量，t = 现金流发生时间（年），y_i 是 t_i 时间的简单贴现率，$B = \sum_{i=1}^{n} \frac{C_i}{(1 + y_i)^{t_i}}$ 是债券价格.[12] 在到期日时，现金流包括本金和支付的利息. 修正久期收益[13] 是久期除以（$1+1/YTM$），其中 YTM 是债券的到期收益率. 然而，用于计算互换的修正久期收益的 YTM 是面值互换利率（例如，使互换的现值为 0 的互换利率）.

为了计算互换的久期（修正久期），我们计算修正久期的看多和看空部分. 固定部分的久期是现金流到期日的现值加权平均，而流动部分的久期是流动部分下一次重置的时间. 久期的两

边同除 1/(1+*YTM*) 得到的就是修正久期. 修正久期的计算方式是：

$$修正久期 = \frac{\text{Leg DV01}}{\text{Leg PV}} * 10\,000 \tag{1.5}$$

其中 DV01（有的时候记为 PV01）是变动一个基点的美元价值[14]，PV 是当前值，这个计算值永远是正数. 另外，修正久期的等价计算方法是：

$$修正久期 = \frac{\text{Leg DV01}}{(名义价值 + 市场价值)} * 10\,000 \tag{1.6}$$

其中市场价值是 Leg PV，这种可替代的计算方式适合于 Leg PV 不包括最终的面值交易. 然而 Leg DV01 一定包括名义久期的变化以绘出一个类似债券久期，因此 Leg DV01 一定包括名义价值.

两个部分修正久期之间的差异——净现值，用来计算互换的修正久期：

$$互换的修正久期 = 收入部分的修正久期 - 支付部分的修正久期 \tag{1.7}$$

注意，如果互换的市场价值（Leg PV）等于零，则久期是可以计算的. 对于一个市场价值为零的面值互换，它的久期和修正久期相等.

对于未来的一个互换（例如一个在 7 年后开始的 10 年期互换），计算贴现互换的修正久期时需要通过一个因子来调整式（1.7），这个因子接近互换开始时的贴现因子. 对于这个贴现因子，一种最佳近似是互换流动部分（包括最终名义变化）的净现值除以交易的名义价值：

$$贴现互换修正久期 = 互换的修正久期 \times 流动部分\ \text{PV}/名义价值 \tag{1.8}$$

注意，修正久期收益（假定统一贴现率）和式（1.5）或式（1.6）中的 DV01 之间的定量差异是由于不同的影响因子，如收益率曲线、互换价值偏离面值、互换开始时间等. 对于向上的贴现率曲线，（到期时）最大现金流的 PV 对久期的影响最大，因为在到期的时候收益率低于市场的贴现率，修正久期收益率的价值高于 DV01 修正久期的价值. 向下的贴现率曲线对久期的有相反的影响，因为在到期的时候收益率比贴现率要高.

由于票面利率与贴现率的不同，导致净现值是不同的. 票面利率的净现值在债券价值（PV）中占有最大比重. 对于 DV01 修正久期，当收益率曲线移动一个基点时，票面利率的现值将减少得更多. 对于一个未来开始的互换，其对应向上的收益率曲线，修正久期收益是两个符号相反的债券修正久期收益之和——一个看多头寸从今天开始，在 T_2 年到期，一个看空头寸从今天开始，在 T_1 年到期，这里 $0<T_1<T_2$，现金流影响收益率发生在 T_1+1 年到 T_2 年之间，在曲线的末端当收益率比贴现率低时贴现. 因此，收益修正久期要高于 DV01 修正久期.

对债券投资组合进行套期保值

考虑表 1.8 中的债券组合的例子. [15]

在这个债券组合中，2011 年 8 月到期 5%的 10 年期活跃债券，2010 年 2 月到期 6.5%的 10 年期 CBOT 国债期货. 表 1.8 中 11 个企业债利率范围为 6.15%到 9.375%，到期从 2007 年 8 月到 2011 年 8 月. 根据美国的级别，这些投资的级别在标准普尔 BBB_- 与 AA_- 之间. 美国标准是 BB_+.

表 1.8 中显示了平均权重久期，考虑整个债券组合，国债部分 262 34 万美元的久期为 6.56 年，436 25 万美元的企业债的久期为 5.39 年，698 59 万美元组合债券的久期为 5.83 年.

不同票面利率和成熟期的固定收益产品对收益变化反应是不同的，这个对收益变化的价格敏感性可以用一个基点的美元价值（DV01）来衡量. 为了得到单独的固定收益的 DV01，给定一个修正久期和全部价格，解决下列问题：

表 1.8

发行机构	息票	期限	到期收益	修正久期（年）	DV01	票额	全价	标准普尔信用评级
国债	6.5	2/15/10	4.55	6.56	0.074 8	67	76 387	
国债部分	5.625	5/15/08	4.64	5.48	0.058 8	92	98 762	
国债部分	5	8/15/11	4.57	7.77	0.080 7	84	87 192	
国债部分				6.56			262 341	
时代华纳公司	8.18	8/15/07	5.47	4.72	0.054 0	82	93 710	BBB+
德州公用事业公司	6.375	1/1/08	6.19	5.06	0.051 7	30	30 678	BBB
罗克韦尔国际公司	6.15	1/15/08	6.14	5.13	0.052 1	52	52 837	A
泛美公司	9.375	3/1/08	6.34	4.93	0.057 3	15	17 445	AA−
海岸公司	6.5	6/1/08	6.76	5.25	0.052 8	30	30 153	BBB
联合航空	6.831	9/1/08	5.99	5.51	0.057 9	38	39 949	A−
伯灵顿北方圣特菲公司	7.34	9/24/08	5.67	5.36	0.060 6	3	3 390	A+
美国新闻集团	7.375	10/17/08	6.56	5.34	0.057 5	30	32 292	BBB−
利顿工业	8	10/15/09	5.70	5.81	0.064 7	63	66 834	BBB−
美国标准公司	7.625	2/15/10	7.59	6.10	0.061 5	41	41 332	BB+
开拓重工	9.375	8/15/11	6.01	6.79	0.085 3	22	27 628	A+
企业绩				5.39			436 248	
投资组合				5.83			698 589	

资料来源：Board of Trade of the City of Chicago，Inc.，© 2001.

$$\text{DV01}=\frac{(久期/100)*全价}{100}$$

例如，利用上述公式与表 1.8 中的价格、久期等数据，我们可以看到一个价值 98 762 000 美元的国债持仓，该国债为 2008 年 5 月到期，收益率为 5.625%，久期为 5.48，DV01 为 54 122美元，而另一个国债品种的持仓价值则为 87 192 000 美元，2011 年 8 月到期，收益率为 5%，久期为 7.77，DV01 为 67 748 美元. 这里 DV01 是指市场收益每上升一个基点，就会使收益率为 5.625%的 2008 年 5 月到期的国债价值下降 54 122 美元；而同样市场收益每上升一个基点，收益率为 5%的 2011 年 8 月到期的国债价值则会下降 67 748 美元，而后者的总持仓价值更小一些. 显然，5%的 2011 年 8 月到期的国债相对于 5.625%的 2008 年 5 月到期的国债来说，对收益变化的敏感度更高.

投资组合的价值与通货膨胀率、利率和信用风险等因素相关，后者可能导致持仓价值出现急剧变化，并使投资者难以以合适的价格进行减持平仓. 对冲仓位的构建首先是要求期货仓位与相应的现货仓位进行匹配，然后还要求期货仓位能够进一步对冲掉相应的现货仓位因市场利率变化所产生的价格波动. 针对一个当时 DV01 为 72.50 美元的 10 年国债期货合约（同期 10 年互换期货的 DV01 为 77.00 美元），我们可以计算出最佳的对冲比例，对 5.625%的 2008 年

5 月国债持仓卖出 747 张合约，对 5%的 2011 年 8 月国债持仓卖出 934 张合约：

$$\frac{2008\text{ 年 5 月 } 5.625\%\text{ 的 DV01}}{\text{期货 DV01}} = \frac{54\,122}{72.50} = 747\text{ 张合约}$$

$$\frac{2011\text{ 年 8 月 } 5\%\text{ 的 DV01}}{\text{期货 DV01}} = \frac{67\,748}{72.50} = 934\text{ 张合约}$$

为了构建投资组合的对冲仓位，我们可以利用国债、企业债或全部持仓的加权平均久期，然后利用各有关资产类别全价来计算 DV01. 表 1.9 显示基于表 1.8 的数据，国债类别的 DV01 是 172 096 美元，企业债类别的 DV01 是 235 138 美元，整个投资组合的 DV01 为 407 277 美元.（注意由于四舍五入，投资组合的总 DV01 并不等于两个组合类别 DV01 的和）

表 1.9

		DV01（$）
国债部分		172 096
企业绩部分		235 138
投资组合		402 277
10 年期国债期货		72.50
10 年期互换期货		77.00
对冲比率		
对冲		对冲比率
1	10 年期中期国债期货投资组合	5 618
2	互换期货投资组合	5 289
3	国债部分的 10 年期中期国债期货	2 374
4	企业绩的互换期货	3 054

资料来源：Board of Trade of the City of Chicago，Inc.，© 2001.

将这个 DV01 值代入对冲比率公式，得出最佳对冲比率. 利用国债期货对冲整个投资组合将需要 5 618 张空头合约：

$$\frac{\text{投资组合 DV01 } 407\,277}{10\text{ 年国债期货 DV01 } 72.50} = 5\,618\text{ 张合约}$$

同样，利用互换期货将需要 5 289 张 10 年期互换期货空头合约对冲整个投资组合：

$$\frac{\text{投资组合 DV01 } 407\,277}{10\text{ 年互换期货 DV01 } 77.00} = 5\,289\text{ 张合约}$$

最后，利用国债期货对冲国债资产、互换期货对冲企业债资产的最优对冲策略需要卖出 2 375张 10 年国债期货合约、3 054 张 10 年互换期货合约：

$$\frac{\text{国债资产 DV01 } 172\,096}{10\text{ 年国债期货 DV01 } 72.50} = 2\,375\text{ 张合约}$$

$$\frac{\text{企业债资产 DV01 } 235\,138}{10\text{ 年互换期货 DV01 } 72.50} = 3\,054\text{ 张合约}$$

必须注意到，这些对冲本身是静态的——它们只能应用于某个特定时点下的市场数据环境. 由于国债、企业债收益率，以及互换利率都会改变 DV01，这些对冲头寸需要持续地进行监控，并根据利率的变化情况、组合的风险暴露情况进行再平衡.

对冲效果与投资组合表现可以通过对市场收益率变化的情景分析与预测分析来衡量. 在 2001 年夏季期间，10 年期国债互换利率信用利差上升了 40 个基点. 企业债收益率基本上与互换利率呈同向运动. 当国债收益率上升 20 个基点时，互换利率与企业债收益率仍然将会上升 60 个基点，给投资组合带来大约 1 755 万美元的损失. 表 1.10 显示了通过 DV01 预测出国债资产类别将会损失 3 441 920 美元，企业债资产类别将会损失 14 108 280 美元. 国债期货对冲（根据预测的 72.50 美元 DV01）将会反映国债收益率的 20 个基点变化，带来 8 146 000

美元的收益（见表 1.10 中的情景 1）.根据这个情景分析，由于对冲将不完全产生 9 404 100 美元的损失，这里的对冲不完全是指利用国债期货对冲只能抵消整个投资组合不到一半的损失.

另一种方式是，利用互换期货来对冲整个投资组合.这种策略的优势是能够对变化更加剧烈的互换利率与企业债收益率作出反应.然而，正如表 1.10 情景 2 所示，根据有关的 DV01 预测，这个对冲策略将在期货头寸上产生远大于投资组合总损失的收益.如表 1.10 情景 3 所示，根据有关 DV01 预测，两个对冲部位将产生 17 551 780 美元的期货收益，带来相对最小的对冲误差，也就是 1 580 美元（这相当于小于 1 个基点，远远小于正常的交易价差，因此在实践意义上是一个较好的对冲策略）.因此，情景 3 看起来比前两种对冲策略的效果都要好一些.这也就突出了利用精确或近似（高度相关性）的对冲工具来对冲各个不同资产类别策略的重要性：利用 10 年期国债对冲国债类别资产，利用 10 年期互换期货对冲企业债类别资产.

表 1.10

相关投资组合的结果				
部分	DV01（美元）	收益变化（bps）	合约数量	收益/损失（美元）
国债	172 096	20		−3 441 920
企业债	235 138	60		−14 108 280
投资组合				−17 550 200
情景 1——以国债期货对冲全部的投资组合				
部分	DV01（美元）	收益变化（bps）	合约数量	收益/损失（美元）
10 年期中期国债	72.05	20	5 618	8 146 100
投资组合				−17 550 200
错误的对冲方式				−9 404 100
情景 2——以互换期货对冲全部的投资组合				
部分	DV01（美元）	收益变化（bps）	合约数量	收益/损失（美元）
10 年期互换	77.00	60	5 289	24 435 180
投资组合				−17 550 200
错误的对冲方式				6 884 980
情景 3——以国债期货对冲国债部分，以互换期货对冲企业债部分				
部分	DV01（美元）	收益变化（bps）	合约数量	收益/损失（美元）
10 年期中期国债	72.50	20	2 374	3 442 300
10 年期互换	77.00	60	3 054	14 109 480
全部对冲结果				17 550 780
投资组合				−17 550 200
投资组合搭配不当				1 380
10 年期中期国债搭配不当				380
企业绩互换搭配不当				1 200

资料来源：Board of Trade of the City of Chicago，Inc. ,© 2001.

与情景 3 中较小的对冲误差相比，情景 1 和情景 2 出现的较大偏差可以用基差风险来解释.这里基差风险是指用单一对冲工具无法完全捕捉由多种类别资产构成的固定收益投资组合

变化所产生的风险. 10 年期国债期货与企业债现货之间的相关性不如 10 年互换期货. 换言之，互换与企业债之间的相关性比国债与企业债之间的相关性要高. 举例来说，用表 1.8 中的泛美公司 9.375%的 2008 年 3 月到期的企业债作为样本，分别进行企业债与互换之间、企业债与国债之间的回归分析. 企业债与互换之间的相关性系数 R^2 为 0.9134，而企业债与国债之间的 R^2 为 0.7966，如图 1.1 所示.

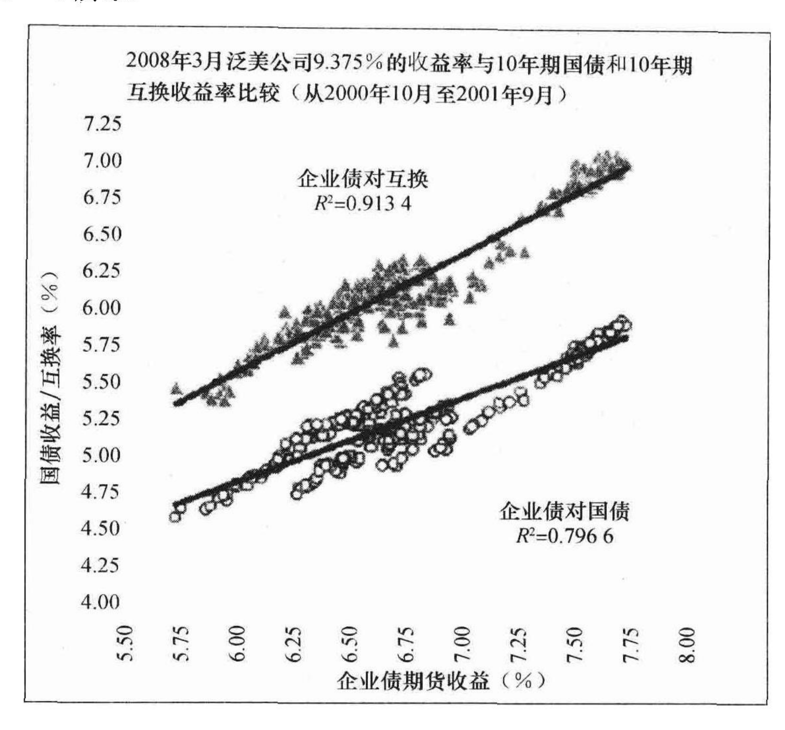


图 1.1

企业债与互换之间的相关性系数 R^2 为 0.913 4 意味着互换利率的变异性代表了 91.34%的企业债收益率的变异性. 相对应的，企业债与国债之间的相关性系数 R^2 为 0.7966 意味着国债利率的变异性代表了不到 80%的企业债收益率的变异性.

1.6 利率期限结构

期限结构模型是固定收益证券及衍生品定价的必要工具. 期限结构模型分析一组证券的价格或收益率之间的定量关系，这组证券的区别仅仅是其现金流发生的时间不同，或者是到期时间不同. 期限结构模型描述了利率随时间变化而变化的情形. 期限结构模型最典型的就是描述平价（par-coupon）收益率关系，尽管通常来说，期限结构模型也可以是贴现函数、即期收益曲线或其他某种价格-收益率关系.

用以定义一个期限结构的一组证券称为*基准组*（reference set），后者可以是一组美国国债、政府代理机构债券、以前发行的国债、利率互换或者 A 级企业债，等等. n 年的零息收益率（也称为 n 年即期利率）是某项投资从今天起一直持续到 n 年的总收益率. 由于平价收益率

曲线在国债市场的广泛应用，许多类别的资产市场都以国债市场的基准组来定义[16]. 例如，“政府代理机构债券的基准组就是当前发行的国债收益率的利差，因此 5 年期政府代理机构债券的定价就是在当前 5 年期国债收益率的基础上增加 15 个基点.”[17] 其他重要的收益率曲线包括远期利率曲线与互换利率曲线. 远期利率（未来的即期利率）就是从当前即期利率里面隐含的未来一段时间的利率. 互换利率曲线则是针对某个固定到期时间的互换而言，收到比如 3 个月 LIBOR浮动利息的一方所要付出的固定利率.

期限结构有三种典型的模式.（息票）收益率曲线是付息债券的到期收益率结构. 零息收益率曲线或即期利率曲线是零息债券的贴现率结构. 远期利率曲线是零息债券的贴现率中隐含的远期利率结构.

1.7 自举分析模型

为在即期利率无法直接观察的情况下构造零息债券收益率曲线，可以使用*自举分析方法*（bootstrap method）从可观察的付息债券价格、互换利率，以及利率期货中分解出零息债券收益率曲线. 自举分析方法，作为一种从先期算得的即期利率和可观察的债券价格中分解即期利率的数量化方法，可以用来构建贴现率曲线. 定义 y_n 为 n 个时限到期债券的收益率，定义 d_n 为 n 个时限到期债券的即期利率（即贴现率）. 自举分析方法的思路就是基于一个付息债券可以被拆解为一组零息债券的组合：

$$\begin{aligned} P_n &= \frac{c/m}{1+y_n/2} + \frac{c/m}{(1+y_n/2)^2} + \cdots + \frac{F+c/m}{(1+y_n/2)^n} \\ &= \frac{c/m}{1+d_n/2} + \frac{c/m}{(1+d_n/2)^2} + \cdots + \frac{F+c/m}{(1+d_n/2)^n} = Z_1 + Z_2 + \cdots + Z_n \end{aligned}$$

对债券收益率曲线的自举分析标准流程称为*递归剥离*（recursive stripping），如下所述：

1. 获得美国国债（或欧洲美元期货）的当前价格.
2. 计算第一个时限的零息债券收益率 z_1.
3. 利用第二个时限的债券价格与 z_1 来计算 z_2.
4. 利用第三个时限的债券价格与 z_1、z_2 来计算 z_3.
5. 继续下去，直到所有的即期利率都已经计算出来.

自举分析就是利用隐含的套利理论，通过已知一个时间节点的数据来推算其他未知各时间节点的数据. 任何付息债券都可以用一组最小化现金流与风险特征的零息债券来定价. 通过对每个理论上的零息债券到期收益率进行作图，直到投资的期限，就可以得到一条理论上的零息利率曲线.

零息即期利率曲线的典型报价方式是以债券同等利率的方式来进行的.

利用自举分析方法可以获得零息利率曲线，后者又可以用来构建贴现率曲线，来对现金流进行贴现. 构建这些曲线通常是从输入一组新发行国债以及部分选出的旧发行国债开始. 所有的现金流都用来构建即期利率曲线，而且各个息票到期日之间的利率呈线性分布.

对于每一个债券到期日，可以通过所有更短时间到期债券的收益率来确定这个债券的所有现金流贴现，后者的和就确定了当前债券的到期收益率. 举例来说，为了确定一个 n 年到期债券的到期收益率，首先需要计算 d_n，而后者又是由到期时间分别为 $i=1$，…，$n-1$ 的一组债

券所决定的.这种方法就叫做自举方法，因为它是通过自上而下，或者通过之前算得的利率来计算最后的利率：

$$P_n = \sum_{i=1}^{n-1} \frac{C_n}{(1+d_i)^i} + \frac{C_n + FV}{(1+d_n/2)^n}$$

对于一个零息债券来说，可以通过每年复利的方式累计各个即期贴现率来算得贴现率 d_c，[18]如下公式：

$$d_c = n\ln\left(1 + \frac{c_n}{n}\right)$$

考虑表 1.11 中所示的国债与债券，假设 5 年之内每 6 个月到期的债券价格都可以获得.

由于新发行国债其更好的流动性，以及交易价格更加接近于平价，同时也使得与溢价、折价有关的税费问题最小化，因此当计算即期利率时，通常使用新发行国债.但是，新发行国债的到期期限只有 0.5 年、1 年、2 年、3 年、5 年和 10 年.在利用这些数据计算即期利率的方法中，最常用的就是指数样条[19].然而，递归剥离也很常用.假设到期面值为 100.

表 1.11

债券价格	年 息	半年期	期限（年）	期间票息
102.296 9	6.125	1	0.5	3.062 5
104.046 9	6.25	2	1.0	3.125
104.000 0	5.25	3	1.5	2.625
103.546 9	4.75	4	2.0	2.375
109.515 6	7.25	5	2.5	3.625
111.171 9	7.5	6	3.0	3.750
122.484 4	10.75	7	3.5	5.375
119.609 4	9.375	8	4.0	4.687
111.328 1	7.0	9	4.5	3.500
108.703 1	6.25	10	5.0	3.125

通过自举分析来获得即期利率曲线，首先计算 0.5 年到期的价格为 102.27 美元的债券的贴现即期利率：

$$102.27 = \frac{(3.0625 + 100)}{(1 + d_1/2)}$$

也就是计算 d_1：

$$d_1 = (103.0625/102.27 - 1) * 2 = 0.014968$$

计算半年期 1 的贴现系数：

$$DiscountSum1 = 1/(1+d_1) = 1/(1.04968) = 0.9853$$

下一步是根据半年期 2 的债券价格来计算第一年的即期利率：

$$104.05 = \frac{3.125}{(1+d_1)} + \frac{103.125}{(1+d_2/2)^2} = 3.125 * DiscountSum1 + 103.125/(1+d_2/2)^2$$

$$= 2 * \left(\left(\frac{\text{期间票息 }2 + \text{面值}}{\text{债券价格 }2 - \text{期间票息 }2 * DiscountSum1}\right)^{1/2} - 1\right)$$

因此，

$$d_2 = 2.1250\%$$

计算半年期 2 的贴现系数：

$$DiscountSum2 = DiscountSum1 + 1/(1+d_2)^2 = 0.9853 + 1/(1.02125)^2 = 1.94407$$

再下一步是根据 1.5 年到期的债券价格来计算 1.5 年的即期利率（第三个半年期）：

$$104 = 2.625 * DiscountSum2 + \frac{102.625}{(1 + d_3/2)^3}$$

也就是计算

$$d_3 = 2 * \left(\left(\frac{\text{期间票息 } 3 + \text{面值}}{\text{债券价格 } 3 - \text{期间票息 } 3 * DiscountSum2} \right)^{1/3} - 1 \right) = 2.482\,2\%$$

因此，一般地，自举分析第 n 个半年期的即期利率公式如下：

$$d_n = 2 * \left(\left(\frac{\text{期间票息}(n) + \text{面值}}{\text{债券价格}(n) - \text{期间票息}(n) * DiscountSum(n-1)} \right)^{1/n} - 1 \right)$$

这里

$$DiscountSum(n-1) = DiscountSum(n-2) + 1/(1 + d_2)^{n-1}$$

持续这样的过程，就可以获得如表 1.12 所示的贴现系数与即期利率数据，以及如图 1.2 所示的利率期限结构.

表 1.12

周　期	贴现系数	即期利率	周　期	贴现系数	即期利率
1	0.985 252 547	1.496 8%	6	5.442 647 715	3.376 6%
2	1.944 068 996	2.125 0%	7	6.227 073 953	3.529 5%
3	2.873 151 16	2.482 2%	8	6.975 043 088	3.696 6%
4	3.766 494 57	2.859 7%	9	7.682 592 791	3.918 7%
5	4.623 301 565	3.139 1%	10	8.359 670 683	3.976 7%

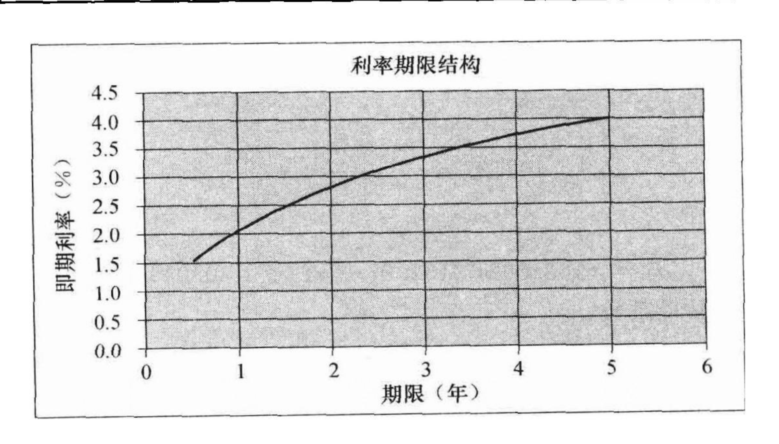


图 1.2　利率期限结构

在实践中，自举分析法需要输入包括短期国债、长期国债、国债期货、欧洲美元期货，以及其他付息债券的价格信息. 有时需要从没有直接价格信息的市场中通过插值分析的方法来获得某个特定到期日的即期利率. 例如，有些债券市场是按照 1 年、3 年、5 年的顺序来排列净收益率的，对于 2 年、4 年的收益率就必须通过插值分析的方法来获得. 插值分析是一种利用如 Newton-Raphson 等数量分析模型来进行试错的方法. 对于付息债券的累计利息计算需要考虑每日盘算惯例带来的影响，这个问题会增加自举分析法的复杂性.

1.8 在 Matlab 中进行自举分析

向 Matlab 中导入数据的方法有三种：(1) 将数据以 ASCII 码的方式直接导入，然后再转换成数量矩阵；(2) 先将数据导入 Excel，然后利用 ExcelLink 将数量矩阵导入 Matlab 工作空间；(3) 利用 Matlab Database Toolbox 从 ODBC-compliant 数据库中导入数据.

Matlab 中的 Financial Toolbox 提供了两个自举分析函数：zbtprice，根据付息债券价格来自举分析零息利率曲线；zbtyield，根据付息债券收益率来自举分析零息利率曲线. 假设我们需要利用图 1.1 中的企业债的收益率曲线数据来自举分析出债券各到期日的零息利率. 假设交割时间为 2006 年 1 月 2 日.

```
% Bonds = [Maturity CouponRate FaceValue]
Bonds = [datenum('15-Aug-2007')  0.08180   100;
         datenum('01-Jan-2008')  0.06375   100;
         datenum('15-Jan-2008')  0.06150   100;
         datenum('01-Mar-2008')  0.09375   100;
         datenum('01-Jun-2008')  0.06500   100;
         datenum('01-Sep-2008')  0.06831   100;
         datenum('24-Sep-2008')  0.07340   100;
         datenum('17-Oct-2008')  0.07375   100;
         datenum('15-Oct-2009')  0.08000   100;
         datenum('15-Feb-2010')  0.07625   100;
         datenum('15-Aug-2011')  0.09375   100];

Yields = [0.0547; 0.0619; 0.0604; 0.0634; 0.0676; 0.0599; 0.0567;
          0.0656; 0.0670; 0.0750; 0.0601];

Prices = [114.28; 102.26; 101.61; 116.30; 100.51; 105.13; 113.00;
          107.64; 111.39; 100.81; 125.58];

Settle = datenum('02-Jan-2006');

[ZeroRates, CurveDates] = zbtyield(Bonds,Yields,Settle)

ZeroRates =

    0.0547
    0.0623
    0.0607
    0.0641
    0.0683
    0.0600
    0.0565
    0.0661
    0.0676
    0.0769
    0.0586

datestr(CurveDates) =

15-Aug-2007
01-Jan-2008
15-Jan-2008
```

```
01-Mar-2008
01-Jun-2008
01-Sep-2008
24-Sep-2008
17-Oct-2008
15-Oct-2009
15-Feb-2010
15-Aug-2011
```

也可以利用债券价格来进行自举分析：

```
    [ZeroRates] = zbtprices(Bonds,Prices,Settle);

ZeroRates =

    0.0251
    0.0581
    0.0582
    0.0400
    0.0665
    0.0554
    0.0362
    0.0533
    0.0548
    0.0810
    0.0469
```

需要注意的是，这些零息利率并不是无风险贴现率，因为企业债并非无风险金融工具. 为了计算无风险零息利率，应该采用短期国债、中期国债，以及长期国债.

1.9 在 Excel 中进行自举分析

考虑图 1.3 所示的 Excel 工作表（ZC. xls），其中一个工作表命名为 Bootstrap，在到期日期间插入零息利率. 为了显示 VB 代码，点击 Tools>Macro>Visual Basic Editor.

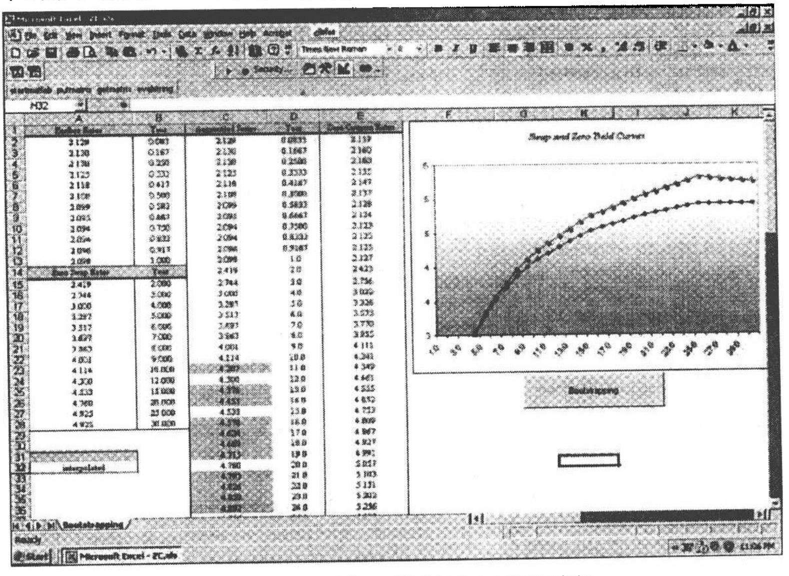


图 1.3 在到期日期间插入零息利率

```
Public ZC(1 To 100) As Double              // stores zero coupon rates
Public swaprate(1 To 100) As Double        // stores swap rates
Public dataswap(1 To 100) As Double
```

VB code in ZC.xls: Sub_interpolation_swap

```
Sub interpolation_swap()

 ReadRates_over10y  'Load swap rates over 10 years

 For i = 1 To 12
   Sheets("Bootstrapping").Cells(i + 1, 3) =
     Sheets("Bootstrapping").Cells(i + 1, 1)
   Sheets("Bootstrapping").Cells(i + 1, 4) =
     Sheets("Bootstrapping").Cells(i + 1, 2)
 Next i

 For i = 1 To 9
   Sheets("Bootstrapping").Cells(i + 13, 3) =
     Sheets("Bootstrapping").Cells(i + 14, 1)
   Sheets("Bootstrapping").Cells(i + 13, 4) =
     Sheets("Bootstrapping").Cells(i + 14, 2)

 Next i

 Sheets("Bootstrapping").Cells(24, 3) = swaprate(12)
 Sheets("Bootstrapping").Cells(27, 3) = swaprate(15)
 Sheets("Bootstrapping").Cells(32, 3) = swaprate(20)
 Sheets("Bootstrapping").Cells(37, 3) = swaprate(25)
 Sheets("Bootstrapping").Cells(42, 3) = swaprate(30)

 'Start the interpolation procedure for each knot

 swaprate(11) = (swaprate(10) + swaprate(12)) * 0.5
 Sheets("Bootstrapping").Cells(23, 3) = swaprate(11)


 For i = 13 To 14
   swaprate(i) = swaprate(12) + ((swaprate(15) - swaprate(12)) *
     ((i - dataswap(2)) / (dataswap(3)
 - dataswap(2))))
   Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
 Next i


 For i = 16 To 19
 swaprate(i) = swaprate(15) + ((swaprate(20) - swaprate(15)) *
   ((i - dataswap(3)) / (dataswap(4) - dataswap(3))))
   Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
 Next i


 For i = 21 To 24
   swaprate(i) = swaprate(20) + ((swaprate(25) - swaprate(20)) *
     ((i - dataswap(4)) / (dataswap(5)
 - dataswap(4))))
   Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
```

```
  Next i


  For i = 26 To 29
  swaprate(i) = swaprate(25) + ((swaprate(30) - swaprate(25)) *
    ((i - dataswap(5)) / (dataswap(6)
  - dataswap(5))))
    Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
  Next i

  End Sub

  Sub ReadRates_over10y()

  Dim i As Integer
  Dim j As Integer
  Sheets("Bootstrapping").Select

  For i = 1 To 6
    dataswap(i) = Sheets("Bootstrapping").Cells(i + 22, 2)
    swaprate(dataswap(i)) = Sheets("Bootstrapping").Cells(i + 22, 1)
  Next i

  End Sub

  Sub ZC_Rates()

  Rates_Load
  ' Uniform the day-count convention between spot and swap rates
    (both 30/360)

  For i = 1 To 12

    ZC(i) = Sheets("Bootstrapping").Cells(i + 1, 3) * 365 / 360

  Next i

  For j = 13 To 41

  dummy_sum = 0

    For i = 1 To j - 12
          dummy_sum = dummy_sum + (swaprate(j) / 100) /
            ((1 + (ZC(11 + i) / 100)) ^ dataswap(11 + i))
    Next i

    ZC(j) = ((((1 + (swaprate(j) / 100)) / (1 - dummy_sum)) ^
      (1 / (dataswap(11 + i)))) - 1) * 100

  Next j

  For i = 1 To 41
    Sheets("Bootstrapping").Cells(i + 1, 5) = ZC(i)
  Next i

End Sub

Sub Rates_Load()
```

```
Dim i As Integer

For i = 1 To 41
  dataswap(i) = Sheets("Bootstrapping").Cells(i + 1, 4)
          swaprate(i) = Sheets("Bootstrapping").Cells(i + 1, 3)
Next i

End Sub
```

1.10 在 Matlab 中计算互换价格的一般方法

Matlab 中的固定收益工具包[21]中包括了用于互换定价、组合对冲的函数. 固定收益工具包中的`liborfloat2fixed`函数，就是用来计算一个浮动换固定互换中与浮动利率同等的固定利率. 这个函数直接给出相对于浮动利率部分现值的固定利率部分现值，无需将两者排列在一起比较浮动与固定部分的不同期限.

以下假设用来进行浮动利率部分的输入：

- LIBOR 利率是季度计算——例如，欧洲美元期货的利率.
- 执行日期是交割日期之后的第三个星期三.
- 所有的交割日期间隔三个月.
- 所有的期限以交割月的第三个星期三开始.
- 所有的期限以开始时后三个月的相同时间结束.
- 复利以 360 天来计算.
- 当远期利率不可获得时，通过插值法估计远期利率.

以下假设用来进行浮动利率部分的输出：

- 在浮动利率输入的基础上建立一个包括票息、付息基准、付息频率的债券.
- 开始日期是一个定价日期——也就是说，是一个签订互换合约的日期.
- 交割可以在开始日期也可以在开始日期之后进行. 如果在之后进行，得到一个远期固定利率合约.
- 执行日期是交割后的第三个星期三，与浮动利率一样.
- 结束日期是数年之后，具体日期与执行日期相同.
- 付息每年进行一次. 频率由债券周期决定.
- 固定利率不进行插值. 固定利率债券的现值与浮动利率的支付额相等.

根据三个月的 LIBOR 数据计算互换的平价固定利率部分，使用以下函数：

```
[FixedSpec, ForwardDates, ForwardRates] =
       liborfloat2fixed(ThreeMonthRates, Settle, Tenor, StartDate,
       Interpolation, ConvexAdj, RateParam, InArrears, Sigma,
       FixedCompound, FixedBasis)
```

对输入的讨论见表 1.13.

表 1.13

参数	说明
ThreeMonthRates	3 个月欧元美元期货数据或远期利率协议数据（远期利率协议规定利率可应用于未来某段时期内的本金总额）. 一个形如［MonthYear IMMQute］的 $n\times3$ 阶矩阵. 浮动利率假设以季度计息，且以实际天数/360 为基准增长

（续）

参　数	说　明
Settle	互换的交割日期，标量
Tenor	互换的期限，标量
StartDate	（可选）表示远期互换估值的参考日期的标量值，这实际上认同了远期互换的估值．默认为 Settle
Interpolation	（可选）当没有欧元美元数据可用时，可以采用插值法来确定远期月利率，默认为“linear”或 1，其余可能的值为“Nearest”或 0，以及“Cubic”或 2
ConvexAdj	（可选）默认值为 0 表示关闭，1 表示开启．其表示是否要求期货/远期凸曲度可调整．当这些数据取自欧元期货数据时，适用于远期利率调整
RateParam	（可选）短期利率模型参数（Hull-White）[a S]，其中短期利率过程如下：$dr=[\theta(t)-ar]dt+Sdz$ 默认为 [0.05　0.015]
InArrears	（可选）默认值为 0 表示关闭，1 表示开启．如果开启，程序将自动对远期利率进行凸曲度调整
Sigma	（可选）全年利率上限波动
FixedCompound	（可选）标量值．对固定一边支付额的复利或频数，也可重置频数．默认为 4，其他值是 1，2 和 12
FixedBasis	（可选）标量值，以固定部分为基准．0＝实际/实际，1＝30/360（SIA，默认），2＝实际/360，3＝实际/365，4＝30/360（PSA），5＝30/360（ISDA），6＝30/360（欧洲），7＝实际/365（日本）

输出如下所示：

FixedBasis 计算远期利率、日期，以及互换固定利率．

FixedSpec 确定了互换固定利率部分的结构：

- Coupon：平价互换利率．
- Settle：开始日期．
- Maturity：结束日期．
- Period：支付频率．
- Basis：复利．

ForwardDates 是对应于 ForwardRates（所有的第三个星期三，每三个月滚动）的日期．首先是 Settle 后的第三个星期三．

ForwardRates 是对应于远期日期的远期利率，以季度复利计算，以实际天数/360 天为基准．为保证输入的正确性，各个期限都向最近的整数舍入．目前交易的间隔是 2 年、5 年和 10 年．函数的浮动利率观察为每个季度交割月的第三个星期三．浮动利率的支付是发生在观察期限内每年的第三个月．

例 3　利用 EDdata.xls 文件作为 liborfloat2fixed 的输入．

```
[EDFutData, textdata] = xlsread('EDdata.xls');
Settle                = datenum('15-Oct-2002');
Tenor                 = 2;

[FixedSpec, ForwardDates, ForwardRates] = ...
liborfloat2fixed(EDFutData, Settle, Tenor)

FixedSpec =
```

```
        Coupon: 0.0222
        Settle: '16-Oct-2002'
      Maturity: '16-Oct-2004'
        Period: 4
         Basis: 1

ForwardDates =

        731505  (16-Oct-2002)
        731596  (15-Jan-2003)
        731687  (16-Apr-2003)
        731778  (16-Jul-2003)
        731869  (15-Oct-2003)
        731967  (21-Jan-2004)
        732058  (21-Apr-2004)
        732149  (21-Jul-2004)

ForwardRates =

      0.0177
      0.0166
      0.0170
      0.0188
      0.0214
      0.0248
      0.0279
      0.0305
```

表 1.14 表示了 2002 年 11 月星期五的欧洲美元数据，我们将之用在 Matlab 中进行收益率曲线与互换利率的计算.

表 1.14

2002 年 11 月星期五的欧洲美元数据

月	年	交割	月	年	交割	月	年	交割	月	年	交割
10	2002	98.21	6	2007	94.88	9	2004	96.745	3	2010	93.82
11	2002	98.26	9	2007	94.74	12	2004	96.515	6	2010	93.755
12	2002	98.3	12	2007	94.595	3	2005	96.33	9	2010	93.7
1	2003	98.3	3	2008	94.48	6	2005	96.135	12	2010	93.645
2	2003	98.31	6	2008	94.375	9	2005	95.955	3	2011	93.61
3	2003	98.275	9	2008	94.28	12	2005	95.78	6	2011	93.56
6	2003	98.12	12	2008	94.185	3	2006	95.63	9	2011	93.515
9	2003	97.87	3	2009	94.1	6	2006	95.465	12	2011	93.47
12	2003	97.575	6	2009	94.005	9	2006	95.315	3	2012	93.445
3	2004	97.26	9	2009	93.925	12	2006	95.16	6	2012	93.41
6	2004	96.98	12	2009	93.865	3	2007	95.025	9	2012	93.39

资料来源：*Matlab*.

利用这些数据，可以通过工具包中的 `liborfloat2fixed` 来计算 1 年、2 年、3 年、4 年、5 年、7 年和 10 年的互换利率. 这个函数仅要求输入欧洲美元数据、交割日期，以及互换的间隔. Matlab 接下来会运行所需要的计算.

为了展示这个函数的运行，首先从 Excel 表格 EDdata.xls 中导入函数.

```
[EDRawData, textdata] = xlsread('EDdata.xls');
```

从第一列中提取月份，从第二列中提取年份.使用的利率为开始与结束时的算术平均利率.

```
Month = EDRawData(:,1);
Year = EDRawData(:,2);
IMMData = (EDRawData(:,3);
EDFutData = [Month, Year, IMMData];
```

接下来，输入当前日期.

```
Settle = datenum('11-Oct-2002');
```

计算两年期的互换利率时，把间隔设置为 2.

```
Tenor = 2;
```

最后，用 liborfloat2fixed 计算互换利率.

```
[FixedSpec, ForwardDates, ForwardRates] =...
     liborfloat2fixed(EDFutData, Settle, Tenor)
```

用默认的设置（季度复利，以 30/360 为基准），以及远期利率（季度复利）数据，Matlab 得到的平价互换利率为 2.23%，而根据美国联邦储备银行公布的表 H15（http://www.federalreserve.gov/releases/h15/update/）中经纪商的平均报价为 2.17%.

```
FixedSpec =

       Coupon: 0.0223
       Settle: '16-Oct-2002'
     Maturity: '16-Oct-2004'
       Period: 4
        Basis: 1

datestr(ForwardDates) =

      731505   (16-Oct-2002)
      731596   (15-Jan-2003)
      731687   (16-Apr-2003)
      731778   (16-Jul-2003)
      731869   (15-Oct-2003)
      731967   (21-Jan-2004)
      732058   (21-Apr-2004)
      732149   (21-Jul-2004)

ForwardRates =

    0.0179
    0.0170
    0.0177
    0.0196
    0.0222
    0.0255
    0.0285
    0.0311
```

在 FixedSpec 的输出中，注意互换利率实际上是从 2002 年 10 月第三个星期三（2002 年 10 月 16 日）开始，原始交割日（2002 年 10 月 11 日）之后的第 5 天.然而，在互换起始日期并不影响互换定价方式与互换存续期限前提下，这仍然是交割日互换利率的最佳替代.

Hull 与 White 建议的修正方法通过调整凸度的方法进一步改进了 liborfloat2fixed 函数的输出结果（见 Hull 的《期权、期货和其他衍生品》第 4 版）. 对于一个较长的互换——比如 5 年或以上——这种修正效果将是显著的.

修正过程需要以下参数：

- StartDate，与输入 Settle（默认）相同，将一个空的矩阵（[]）作为输入.
- ConvexAdj，告诉 liborfloat2fixed 函数，运行修正.
- RateParam，提供了参数 a 与 s，作为 Hull-White 短期过程的参数.
- 可选的参数 InArrears 与 Simga，可用一个空的矩阵（[]）来接受 Matlab 的默认数值.
- FixedCompound，可以提供将默认的每季度复利转换为每半年复利，以便于与美国联邦储备银行公布的统计数据进行比较.

```
StartDate = [];
Interpolation = [];
ConvexAdj = 1;
RateParam = [0.03; 0.017];
FixedCompound = 2;
[FixedSpec, ForwardDates, ForwardRates] =...
liborfloat2fixed(EDFutData, Settle, Tenor, StartDate, Interpolation,
ConvexAdj, RateParam, [], [], FixedCompound)
```

得到 2.21% 的两年互换利率，与公布的相应日期的互换利率接近. 作为对比，表 1.15 总结了 1 年、3 年、5 年、7 年以及 10 年的互换利率（凸性调整与无调整）.

表 1.15

2002 年 10 月 11 日星期五，经过测算的市场平均互换率数据

互换跨度（年）	调整前	调整后	表格 H15	调整错误（基点）	互换跨度（年）	调整前	调整后	表格 H15	调整错误（基点）
1	1.80%	1.79%	1.80%	−1	5	3.50%	3.37%	3.36%	+1
2	2.24%	2.21%	2.22%	−1	7	4.16%	3.92%	3.89%	+3
3	2.70%	2.66%	2.66%	0	10	4.87%	4.42%	4.39%	+3
4	3.12%	3.03%	3.04%	−1					

资料来源：*Matlab.*

用 liborduration 函数来计算一个以 LIBOR 为基础的利率互换久期：

```
[PayFixDuration GetFixDuration] = liborduration(SwapFixRate, Tenor,Settle)
```

输入的讨论见表 1.16.

表 1.16

参数	说明
SwapFixRate	互换固定利率的标量或列向量（以十进制方式表示）
Tenor	表明互换期限的标量或列向量（年），小数部分向上舍入
Settle	交割日的标量或列向量

输出的讨论如下：

- PayFixDuration 是修正久期，以年计算，当输入互换的支付固定部分时实现.
- GetFixDuration 是修正久期，以年计算，当输入互换的收到固定部分时实现.

例 4 已知如下数据：

```
SwapFixRate = 0.0383;
Tenor = 7;
Settle = datenum('11-Oct-2002');
```

计算互换久期.

```
[PayFixDuration GetFixDuration] = liborduration(SwapFixRate,...
    Tenor, Settle)

PayFixDuration =

   -4.7567

GetFixDuration =

    4.7567
```

在 Matlab 中，可用 Black-Derman-Toy(BDT) 树或 HJM 树对互换进行估价. 程序代码如下：

```
[Price, PriceTree, CFTree, SwapRate] = swapbybdt(BDTTree, LegRate,
   Settle, Maturity, LegReset, Basis, Principal, LegType, Options)
```

输入数据见表 1.17.

表 1.17

参数	说明
BDTTree	利率树结构通过 bdttree 创建
LegRate	NINST×2 矩阵，每行定义为：[CouponRate Spread] 或 [Spread CouponRate]. CouponRate 是十进制年利率. Spread 是超过参考利率的那部分基点. 第一列代表收入部分，第二列代表支出部分
Settle	交割日期. NINST×1 向量，是一系列的日期数字或字符串. Settle 必须在 Maturity 前
Maturity	到期日，NINST×1 向量数据代表了每个互换的到期日
LegReset	（可选）NINST×2 矩阵代表了每个互换每年重置的频率. 默认为 [1 1]
Basis	（可选）NINST×1 向量代表了输入年化利率树时的基准点. 默认值为 0（实际/实际）
Principal	（可选）NINST 向量代表名义本金数量. 默认=100
LegType	（可选）NINST×2 矩阵. 每行代表一个结构，每列表明了相应部分是否固定（1）或浮动（0）. 这个矩阵对 LegRate 值给出了定义. 每个结构的默认值是 [1 0]
Options	（可选）用 derivset 创建衍生品价格期权结构

输出如下：

- Price 是互换在 0 时刻、NINST(number of instruments) 为 1 的数值.
- PriceTree 是互换价值向量三叉树定价中每个节点的价值.
- CFTree 是互换现金流向量三叉树定价中每个节点的价值.
- SwapRate 是在 0 时刻，使互换价值为 0 的 NINST 为 1 的向量利率. 这个利率是当其在 LegRate 的固定部分中为 NaN 时用来计算互换价值的. NaN 是 CouponRate 没有设定为 NaN 时作为 SwapRate 的过渡.

例 5 给一个收到固定利率、支付浮动利率的互换定价，每年进行支付，名义价值为 100 万美元. 其他参数为：

- 固定部分的票息率：0.15 (15%).
- 浮动部分的基差：10 个基点.
- 互换开始日期：2000 年 1 月 1 日.

- 互换到期日期：2003 年 1 月 1 日.

基于以上信息，设立所需的参数，并建立 LegRate、LegType 和 LegReset 矩阵.

```
Settle = '01-Jan-2000';
Maturity = '01-Jan-2003';
Basis = 0;
Principal = 1000000;
LegRate = [0.15 10]; % [CouponRate Spread]
LegType = [1 0]; % [Fixed Float]
LegReset = [1 1]; % Payments once per year
```

利用在 MAT 文件 deriv.mat 中的 BDTTree 来给互换定价. BDTTree 包含时间以及远期利率等信息，可用以定价.

```
load deriv;
```

利用 swapbybdt 来给互换定价.

```
Price = swapbybdt(BDTTree, LegRate, Settle, Maturity,...
     LegReset, Basis, Principal, LegType)

Price = 73032
```

例 6 利用以上数据，计算互换利率，也就是当 0 时刻互换价值为 0 时的固定部分息票率.

```
LegRate = [NaN 20];

[Price, PriceTree, CFTree, SwapRate] = swapbybdt(BDTTree,...
    LegRate, Settle, Maturity, LegReset, Basis, Principal, LegType)

Price =

 -2.8422e-014

PriceTree =

    FinObj: 'BDTPriceTree'
     tObs: [0 1 2 3 4]
     PTree: {1x5 cell}

CFTree =

    FinObj: 'BDTCFTree'
     tObs: [0 1 2 3 4]
     CFTree: {1x5 cell}

SwapRate =
             0.1210
```

使用 HJM 树也可以给互换定价.

```
[Price, PriceTree, CFTree, SwapRate] = swapbyhjm(HJMTree, LegRate,
     Settle, Maturity, LegReset, Basis, Principal, LegType, Options)
```

关于 swapbyhjm 函数的讨论见表 1.18.

表 1.18

参数	说明
HJMTree	用 hjmtree 创建远期利率树结构
LegRate	NINST×2 矩阵，每行定义为：[CouponRate Spread] 或 [Spread CouponRate]. CouponRate 是十进制年利率. Spread 即超过参考利率的那部分基点. 第一列代表收入部分，第二列代表支出部分

(续)

参 数	说 明
Settle	NINST×1 向量是一系列的日期数字或字符串. Settle 必须在 Maturity 之前
Maturity	NINST×1 向量数据代表了每个互换的到期日
LegReset	(可选) NINST×2 矩阵代表了每个互换每年重置的频率. 默认=[1 1]
Basis	(可选) NINST×1 向量代表了输入年化利率树时的基准点. 默认值=0 (实际/实际)
Principal	(可选) NINST×1 向量代表名义本金数量. 默认=100
LegType	(可选) NINST×2 矩阵. 每行代表一个结构, 每列表明了相应部分是否固定 (1) 或浮动 (0). 这个矩阵对 LegRate 值给出了定义. 每个结构的默认值是 [1 0].
Options	(可选) 用 derivset 创建衍生品价格期权结构

每个互换的交割日期都设定为 HJM 树中的 ValuationDate, 互换的 Settle 被忽略. 该函数也可以计算当互换初始价值为 0 时的 SwapRate (固定利率). 为此, 需要将 NaN 设定为 CouponRate.

描述

[Price,PriceTree,CETree,SwapRate]= swapbyhjm(HJMTree,LegRate,Settle,Maturity,LegReset,Basis,Principal,LegType)利用 HJM 利率树来计算互换价值.

Price	NINST×1 向量代表 0 时刻预期的互换价格
PriceTree	以互换价格向量为结点的树结构
CFTree	以互换现金流向量为结点的树结构
SwapRate	NINST×1 向量代表对固定部分的利率, 如在 0 时刻, 互换价格为 0. 当 LegRate 中固定部分的利率指定为 NaN 时, 利用这个利率来计算互换价格. SwapRate 用 NaN 填补, 而其中的 CouponRate 不被设置为 NaN

例 7 给一个收到固定利率、支付浮动利率的互换定价, 每年进行支付, 名义本金为 100 美元. 其他参数为:

- 固定部分的息票率: 0.06 (6%).
- 浮动部分的基差: 20 个基点.
- 互换开始日期: 2000 年 1 月 1 日.
- 互换到期日期: 2003 年 1 月 1 日.

基于以上信息, 设立所需的参数, 并建立 LegRate、LegType 和 LegReset 矩阵.

```
Settle = '01-Jan-2000';
Maturity = '01-Jan-2003';
Basis = 0;
Principal = 100;
LegRate = [0.06 20]; % [CouponRate Spread]
LegType = [1 0]; % [Fixed Float]
LegReset = [1 1]; % Payments once per year
```

利用在 MAT 文件 deriv.mat 中 BJMTree 来给互换定价. BJMTree 包含需要用于定价的时间以及远期利率等信息.

```
load deriv;
```

利用 swapbyhjm 计算互换价格.

```
[Price, PriceTree, CFTree] = swapbyhjm(HJMTree, LegRate,...
     Settle, Maturity, LegReset, Basis, Principal, LegType)

Price =

   3.6923

PriceTree =

    FinObj: 'HJMPriceTree'
      tObs: [0 1 2 3 4]
     PBush: {1x5 cell}

CFTree =

    FinObj: 'HJMCFTree'
      tObs: [0 1 2 3 4]
    CFBush: {[0] [1x1x2 double] [1x2x2 double] ... [1x8 double]}
```

1.11 在 Matlab 中利用期限结构分析为互换定价

本例演示了在金融工具包里的一些期限结构分析函数，特别是演示了如何从付息债券的市场价格中找出隐含的零息（即期，spot）利率曲线与远期利率曲线. 零息利率曲线与远期利率曲线可以用来为利率互换合约进行定价. 在一个利率互换中，参与双方就未来一段时间内的现金流互换达成一致. 其中一方的现金流为在互换合约存续期内基于一个固定的利率来支付，另一方的现金流则与某些可变的利率指标相挂钩. 为一个初始的互换定价，就是要找出互换合约中的固定部分利率. 由互换合约的名义本金中约定的固定利率决定了按时产生的固定现金流. 通常，用远期利率曲线来为利率互换定价，因为互换中的浮动现金流是由一系列远期利率所决定的，而固定现金流的净现值都相等. 因此，利率互换的定价与期限结构分析高度相关.

步骤 1 定义 10 种美国国债的交割日、到期日、票息率，以及市场价格. 这些数据可以为一个 5 年期每 6 个月交换现金流的互换定价. 为简单起见，接受默认的设置，也就是每个月的月底支付，以实际天数除以实际天数计算复利. 为了避免出现累计利息，假设所有的国债都是每半年付息，且在付息当日就完成交割.

```
Settle = datenum('15-Jan-1999');

BondData = {'15-Jul-1999' 0.06000 99.93
            '15-Jan-2000' 0.06125 99.72
            '15-Jul-2000' 0.06375 99.70
            '15-Jan-2001' 0.06500 99.40
            '15-Jul-2001' 0.06875 99.73
            '15-Jan-2002' 0.07000 99.42
            '15-Jul-2002' 0.07250 99.32
            '15-Jan-2003' 0.07375 98.45
            '15-Jul-2003' 0.07500 97.71
            '15-Jan-2004' 0.08000 98.15};
```

BondData 是 Matlab 单元阵列（cell array）的一个例子，通过大括号 {} 来表示.

接下来，为进一步的处理方便，将数据以单元阵列的形式分配给 Maturity、CouponRate 和 Prices 向量.

```
Maturity   = datenum(strvcat(BondData{:,1}));
CouponRate = [BondData{:,2}]';
Prices     = [BondData{:,3}]';
Period     = 2; % semiannual coupons
```

步骤 2 既然数据都已经定义完毕，利用期限结构分析函数 zbtprice 就可以从付息债券的价格中自举得出隐含的零息利率曲线. 隐含的零息利率曲线的意义是一系列的零息债券利率与付息债券价格完全对应，使两者之间不存在套利空间.

```
ZeroRates = zbtprice([Maturity CouponRate], Prices, Settle);
```

保存在 ZeroRates 中的零息曲线是按半年付息债券的形式来报价的（持有期限是 6 个月，利率简单加倍就可以年化）. ZeroRates 中的第一部分是 6 个月的年化收益率，第二部分是 12 个月的年化收益率，以此类推.

步骤 3 通过隐含零息曲线，利用期限结构函数**zero2fwd**得出相应的隐含远期利率序列.

```
ForwardRates = zero2fwd(ZeroRates, Maturity, Settle);
```

保存在 ForwardRates 中的远期利率曲线也是按半年付息债券的形式来报价的. ForwardRates 中的第一部分是从交割日至交割日之后 6 个月之间的年化收益率，第二部分是第 6 个月至第 12 个月之间的年化收益率，以此类推. 隐含远期利率曲线的意义也是与付息债券价格完全对应的，两者之间不存在套利空间. 由于第一个远期利率同时也是零息利率，所以 ZeroRates 与 ForwardRates 的第一部分相同.

步骤 4 既然已经得出零息曲线，利用期限结构函数**zero2disc**就可以将它转换为一系列的贴现因子.

```
DiscountFactors = zero2disc(ZeroRates, Maturity, Settle);
```

步骤 5 利用贴现因子，将从隐含远期利率曲线中得出的一系列浮动现金流进行贴现，得到现值. 对于典型利率互换来说，互换的名义本金在每个支付日期都保持恒定，可以在现值方程中从两边相互抵消. 下一行中假设了名义本金为 1.

```
PresentValue = sum((ForwardRates/Period) .* DiscountFactors);
```

步骤 6 计算互换价格，也就是使得固定部分现金流现值与从隐含远期利率曲线中所得浮动部分现金流现值相等的固定利率. 同样，互换的名义本金可以从两边相互抵消，简单地假设为 1.

```
SwapFixedRate = Period * PresentValue / sum(DiscountFactors);
```

输出如下：

```
Zero Rates  Forward Rates
  0.0614     0.0614
  0.0642     0.0670
  0.0660     0.0695
  0.0684     0.0758
  0.0702     0.0774
  0.0726     0.0846
  0.0754     0.0925
  0.0795     0.1077
  0.0827     0.1089
  0.0868     0.1239

Swap Price (Fixed Rate) = 0.0845
```

所有的利率以小数形式表达. 互换价格为 8.45%，正好是市场做市商报价的中间值.

例 8 一个 9 年期固定换浮动的互换，名义价值为 330 万美元，固定利率为 3.969%，浮动利率基于三个月期美国短期国债，开始日期为 2004 年 3 月 28 日，生效日期为 2004 年 3 月 29 日，到期日为 2013 年 3 月 28 日. 假设固定部分以 30/360 计算，浮动部分以实际天数除以

360 天计算，每季度调整. 图 1.4 显示了 Bloomberg 上显示的互换数据.

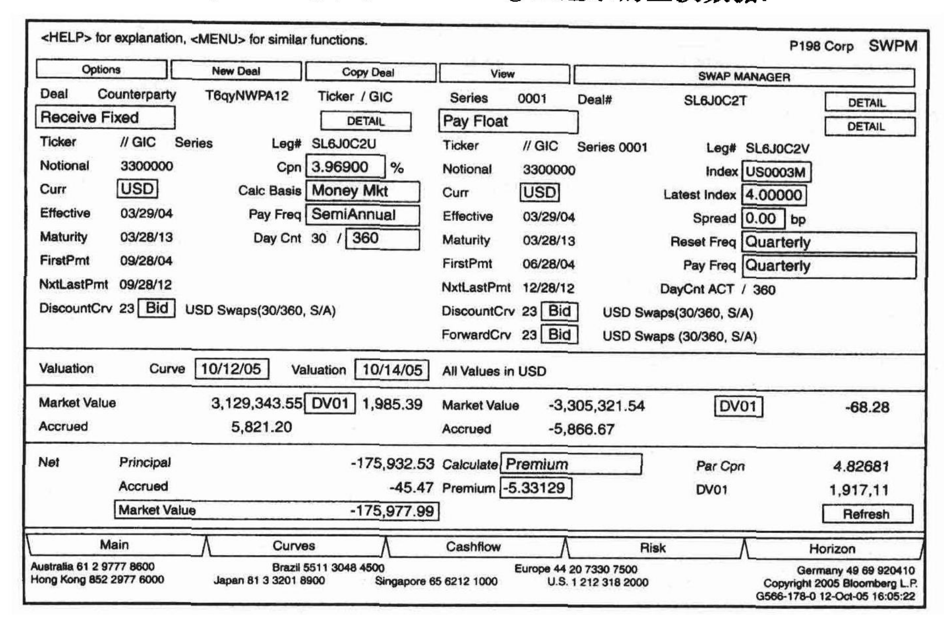


图 1.4

资料来源：Bloomberg.

图 1.5 至图 1.7 显示了互换的现金流支付金额以及支付日期. 图 1.8 提供了互换利率曲线. 图 1.9 显示了对支付部分与收入部分风险的度量、DV01 和久期.

<HELP> for explanation. P235 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

REC FIXED Coupon 3.96900 Frequency S Curr USD Notional 3300000

PAY FLOAT Latest Index 4.00000 + 0.00 bp Reset/Pmnt FreqQ/Q Curr USD Notional 3300000

Net Cashflo Currency USD EXPORT TO EXCEL

Payment Dates	Payments(Rcv)	Payments(Pay)	Net Payments	Discount	Net PV
12/28/2005	0.00	-33366.67	-33366.67	0.991587	-33085.94
03/28/2006	65488.50	-36550.57	28937.93	0.980724	28380.12
06/28/2006	0.00	-38758.48	-38758.48	0.969339	-37570.12
09/28/2006	65488.50	-39287.93	26200.57	0.957935	25098.43
12/28/2006	0.00	-38326.13	-38326.13	0.946937	-36292.42
03/28/2007	65488.50	-38039.88	27448.62	0.936146	25695.90
06/28/2007	0.00	-39213.28	-39213.28	0.925152	-36278.26
09/28/2007	65488.50	-39523.30	25965.20	0.914203	23737.47
12/28/2007	0.00	-39297.44	-39297.44	0.903445	-35503.06
03/28/2008	65488.50	-39531.82	25956.68	0.892750	23172.83
06/30/2008	0.00	-41091.95	-41091.95	0.881770	-36233.66
09/29/2008	65852.32	-39999.08	25853.25	0.871210	22523.61
12/29/2008	0.00	-39094.19	-39094.19	0.861010	-33660.49
TOTAL					-176149.58

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Hong Kong 852 2977 6000 Brazil 5511 3048 4500 Japan 81 3 3201 8900 Singapore 65 6212 1000 Europe 44 20 7330 7500 U.S. 1 212 318 2000 Germany 49 69 920410 Copyright 2005 Bloomberg L.P. G566-178-0 18-Nov-05 15:08:34

图 1.5

资料来源：Bloomberg.

<HELP> for explanation. P235 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

REC FIXED Coupon 3.96900 Frequency S Curr USD Notional 3300000
PAY FLOAT Latest Index 4.00000 + 0.00 bp Reset/Pmnt FreqQ/Q Curr USD Notional 3300000

Net Cashflo Currency USD EXPORT TO EXCEL

Payment Dates	Payments(Rcv)	Payments(Pay)	Net Payments	Discount	Net PV
12/29/2008	0.00	-39094.19	-39094.19	0.861010	-33660.49
03/30/2009	65852.32	-38911.34	26940.99	0.850976	22926.13
06/29/2009	0.00	-38927.60	-38927.60	0.841055	-32740.24
09/28/2009	64760.85	-38938.01	25822.84	0.831247	21465.15
12/28/2009	0.00	-39881.18	-39881.18	0.821321	-32755.23
03/29/2010	65852.32	-40177.71	25674.62	0.811441	20833.45
06/28/2010	0.00	-40274.03	-40274.03	0.801658	-32285.99
09/28/2010	65124.68	-40807.77	24316.91	0.791866	19255.72
12/28/2010	0.00	-40583.21	-40583.21	0.782245	-31746.04
03/28/2011	65488.50	-40243.83	25244.67	0.772821	19509.61
06/28/2011	0.00	-41217.01	-41217.01	0.763287	-31460.42
09/28/2011	65488.50	-41282.22	24206.28	0.753857	18248.07
12/28/2011	0.00	-40707.37	-40707.37	0.744671	-30313.59
TOTAL					-176149.58

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2005 Bloomberg L.P.
G566-178-0 18-Nov-05 15:09:10

图 1.6
资料来源：Bloomberg.

<HELP> for explanation. P235 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

REC FIXED Coupon 3.96900 Frequency S Curr USD Notional 3300000
PAY FLOAT Latest Index 4.00000 + 0.00 bp Reset/Pmnt FreqQ/Q Curr USD Notional 3300000

Net Cashflo Currency USD EXPORT TO EXCEL

Payment Dates	Payments(Rcv)	Payments(Pay)	Net Payments	Discount	Net PV
03/29/2010	65852.32	-40177.71	25674.62	0.811441	20833.45
06/28/2010	0.00	-40274.03	-40274.03	0.801658	-32285.99
09/28/2010	65124.68	-40807.77	24316.91	0.791866	19255.72
12/28/2010	0.00	-40583.21	-40583.21	0.782245	-31746.04
03/28/2011	65488.50	-40243.83	25244.67	0.772821	19509.61
06/28/2011	0.00	-41217.01	-41217.01	0.763287	-31460.42
09/28/2011	65488.50	-41282.22	24206.28	0.753857	18248.07
12/28/2011	0.00	-40707.37	-40707.37	0.744671	-30313.59
03/28/2012	65488.50	-40705.47	24783.03	0.735597	18230.33
06/28/2012	0.00	-41183.89	-41183.89	0.726530	-29921.34
09/28/2012	65488.50	-41205.10	24283.40	0.717570	17425.05
12/28/2012	0.00	-41725.89	-41725.89	0.708611	-29567.41
03/28/2013	3365488.50	-3341534.93	23953.57	0.699803	16762.77
TOTAL					-176149.58

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2005 Bloomberg L.P.
G566-178-0 18-Nov-05 15:09:39

图 1.7
资料来源：Bloomberg.

<HELP> for explanation. P198 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

Curve #23 USD Swaps (30/360,S/A)

Current Market			6 Month −50bp			6 Month +0bp			6 Month +50bp		
#	Mty/Term	Rate	#	Mty/Term	Rate	#	Mty/Term	Rate	#	Mty/Term	Rate
1	1 DY	3.50000	1	1 DY	3.00000	1	1 DY	3.50000	1	1 DY	4.00000
2	2 DY	3.80000	2	2 DY	3.30000	2	2 DY	3.80000	2	2 DY	4.30000
3	1 WK	3.81688	3	1 WK	3.31688	3	1 WK	3.81688	3	1 WK	4.31688
4	2 WK	3.82000	4	2 WK	3.32000	4	2 WK	3.82000	4	2 WK	4.32000
5	3 WK	3.86000	5	3 WK	3.36000	5	3 WK	3.86000	5	3 WK	4.36000
6	1 MO	3.94563	6	1 MO	3.44563	6	1 MO	3.94563	6	1 MO	4.44563
7	2 MO	4.02563	7	2 MO	3.52563	7	2 MO	4.02563	7	2 MO	4.52563
8	3 MO	4.14000	8	3 MO	3.64000	8	3 MO	4.14000	8	3 MO	4.64000
9	4 MO	4.20000	9	4 MO	3.70000	9	4 MO	4.20000	9	4 MO	4.70000
10	5 MO	4.26000	10	5 MO	3.76000	10	5 MO	4.26000	10	5 MO	4.46000
11	6 MO	4.32875	11	6 MO	3.82875	11	6 MO	4.32875	11	6 MO	4.82875
12	7 MO	4.37025	12	7 MO	3.87025	12	7 MO	4.37025	12	7 MO	4.87025

Current Market		6 Month −50bp		6 Month +0bp		6 Month +50bp	
Horizon Curve Date	10/12/05	Horizon Curve Date	04/12/06	Horizon Curve Date	04/12/06	Horizon Curve Date	04/12/06
Horizon Settle Date	10/14/05	Horizon Settle Date	04/18/06	Horizon Settle Date	04/18/06	Horizon Settle Date	04/18/06
GLOBAL CHANGE FIELDS	-----›	From 1 To 34	Shift 0.00	From 1 To 34	Shift 0.00	From 1 To 34	Shift 0.00
Pay Leg PV	-3,305,321.54	Pay Leg PV	-3,331,694.62	Pay Leg PV	-3,308,454.83	Pay Leg PV	-3,305,221.38
Receive Leg PV	3,128,009.55	Receive Leg PV	3,239,101.26	Receive Leg PV	3,143,640.16	Receive Leg PV	3,051,445.05
Net PV	-177,311.99	Net PV	-72,593.36	Net PV	-164,814.67	Net PV	-253,776.32

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000

G566-178-0 12-Oct-05 15:48:54

图 1.8

资料来源：Bloomberg.

<HELP> for explanation. P198 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

Risk Key Rate Risk

Conventional	Receive side	Pay side	Net	Mty/Term	Receive side	Pay side	Net
Risk	6.01	-0.21	5.81	1 DY	-0.00	-0.00	-0.00
DV01	1984.83	-68.28	1916.55	2 DY	0.00	-0.00	-0.00
Modified Duration	6.36	-0.22	6.14	1 WK	0.00	0.00	0.00
				2 WK	0.00	0.00	0.00
				3 WK	0.00	0.00	0.00
				1 MO	0.00	0.00	0.00
				2 MO	0.00	-40.17	-40.17
				3 MO	0.00	-28.12	-28.12
				4 MO	0.00	0.00	0.00
				5 MO	1.70	-0.00	1.70
				Total	1985.56	-68.28	1917.28

Valuation Curve 10/12/05 Valuation 10/14/05 All Values in USD

Market Value	3,128,009.55	DV01	1,984.83	Market Value	-3,305,321.54	DV01	-68.28
Accrued	5,821.20			Accrued	-5,866.67		

Net	Principal	-177,766.53	Calculate	Premium	Par Cpn	4.83338
	Accrued	-45.47	Premium	-5.33129	DV01	1,916.55
	Market Value	-177,311.99				Refresh

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000

G566-178-0 12-Oct-05 15:48:11

图 1.9

资料来源：Bloomberg.

1.12 利用 C++ 程序进行互换定价

为了用 C++ 给固定换浮动互换定价，我们定义一个 Swap 类. Swap 类包含两个部分——

一个是 FloatingLeg 类，作为浮动部分；一个是 FixedLeg 类，作为固定部分，分别代表互换的两个部分.由于支付日期的计算是基于开始日期、生效日期和到期日期，Swap 类需要用到一个 Date 类[22]来进行有关日期的运算. Date 类在文件 datecl. h 中定义，并在全书都被用于计算支付日期.

Swap 类包含以下 Date 数据：

```
Date maturity_;              // swap maturity date
Date fixedAccruedDate_;      // fixed interest accrual date
Date floatAccruedDate_;      // floating interest accrual date
Date effectiveDate_;         // effective date
Date settlementDate_;        // settlement date
Date valuationDate_;         // valuation date
```

Date 的有关计算方法在文件 datecl. cpp 中定义. Swap 类通过不同的线内方法来加进对互换支付金额、净额和价值的计算：

```
double getNotional()         // return notional amount
double calcDV01()            // compute swap DV01
void netPayments()           // net fixed and floating payments
void calcPayDates(Date tradeDate, Date endDate, Date valuation)
  // calculate pay dates
void setDiscountRates(std::map<double,double> rate)  // set discount rates
```

Swap 类利用 FloatLeg 与 FixedLeg 数据类来运行上述这些方法. Swap 类包含用于接收与存储定价所需重要数据的重载构造函数：到期日期、生效日期、交割日期、定价日期、浮动（LIBOR）利率、贴现率、固定互换利率，以及定价类型（收到固定支付浮动还是收到浮动支付固定）. Swap 类的定义为：

SWAP.h

```
#ifndef _SWAP_H__
#define _SWAP_H__

#include "TNT\TNT.h"
#include "datecl.h"
#include <string>
#include <vector>
#include <map>
#define NUM_DATES          100
#define THIRTY              30
#define THREE_SIXTY        360
#define THREE_SIXTY_FIVE   365
#define NOTIONAL           1000000

static std::vector<Date> payDates_;

static double interpolate(double rate1, double rate2, double t1, double t2,
  double x) {
  double dy = rate2 - rate1;
  double dt = t2 - t1;
  double slope = dy/dt;

  return rate1 + slope*x;
}

class FloatingLeg
```

```
{
  public:
    FloatingLeg(std::map<double,double> floatLegRate, double floatLegBasis,
      int payFrequency)
                : floatLegRate_(floatLegRate),floatLegBasis_(floatLegBasis),
                  payFrequency_(payFrequency) {}
    FloatingLeg() {}
    virtual ~FloatingLeg() {}
    inline double calcDuration() {
      double duration = 0.0;
      duration = (double) (payDates_[0] -
        valuationDate_ + 1)/THREE_SIXTY_FIVE;
      return duration;
    }
    inline void setEffectiveDate(Date date) { startDate_ = date; }
    inline void setValuationDate(Date date) { valuationDate_ = date; }
    inline void setNotional(double notional) { notional_ = notional; }
    inline Date getEffectiveDate() { return startDate_; }
    inline void setFrequency(int frequency) { payFrequency_ = frequency; }
    inline void setMaturityDate(Date mat) { maturityDate_ = mat; }
    inline std::vector<double> getPayFloat() { return floatRates; }
    inline void setFloatValue(double value) { value_ = value; }
    inline double getFloatValue() {
        return value_;
    }
    inline void setFloatRate(std::map<double,double> rate) {
          floatLegRate_ = rate;
    }
    inline double calcDV01() {

          double duration = calcDuration();
          double val = getFloatValue();
          double DV = 0.0;

          DV = -(duration*notional_)*((double)1/10000);
          cout << "float DV01 = " << DV << endl;

          return -DV;
    }
    inline double calcModifiedDuration() {

          double val = calcDV01();
          double marketValue = getFloatValue();
          double MD = (val/(notional_ + marketValue))*10000;

          cout << "float modified duration = " << MD << endl;

          return MD;
    }
    inline void calcPayFloat() {

          std::vector<Date>::iterator iter;
          double val = 0;
          double diff = 0.0;
          int diff1 = 0.0;
          int d = 0.0;
          int d1 = 0.0;
          Date dateDiff;
          int cnt = 0;
```

```
        double floatVal = 0.0;

        for (iter = payDates_.begin(); iter != payDates_.end(); iter++)
        {
          d = payDates_[cnt+1] - payDates_[0] + 1;
          d1 = payDates_[cnt+1] - payDates_[cnt] + 1;

          diff = payDates_[cnt+1] - payDates_[0]+1;
          diff = (double) diff/THREE_SIXTY_FIVE;
          floatVal = interpolate(floatLegRate_[floor(diff)],
            floatLegRate_[ceil(diff)],floor(diff),ceil(diff),diff);

          if (payFrequency_ == 1)
                val = notional_*(THIRTY/THREE_SIXTY)*floatVal;
          else
                val = notional_*((double)d1/THREE_SIXTY_FIVE)*floatVal;
          floatRates.push_back(val);
          cnt++;
        }
    }
  private:
    double floatLegBasis_;
        std::vector<double> floatRates;
        std::map<double,double> floatLegRate_;
        std::map<double,double> payfloatLeg_;
        Date startDate_;
        Date maturityDate_;
        Date valuationDate_;
        double value_;
        double spread_;
        double notional_;
        double duration_;
        double accrual_;
        int payFrequency_;
};

class FixedLeg
{
  public:
        FixedLeg(double payfixedLeg, double fixedLegRate,
          double fixedLegBasis, int payFrequency)
      : payfixedLeg_(payfixedLeg), fixedLegRate_(fixedLegRate),
        fixedLegBasis_(fixedLegBasis),
                payFrequency_(payFrequency) {}
        FixedLeg() {}
        virtual ~FixedLeg() {}
        inline double calcDV01() {

          double duration = calcDuration();
          double val = getFixedValue();
          double DV = 0.0;

          DV = (duration*notional_)*((double)1/10000);
          cout << "fixed DV01 = " << DV << endl;

          return DV;
        }
        inline void setEffectiveDate(Date date) { effectiveDate_ = date; }
```

```
inline double calcDuration() {

  double sum = 0;
  double val = getFixedValue();
  double duration = 0.0;
  double dis = 0.0;

  for (int i = 0; i < payfixedLeg_.size(); i++)
     sum = sum + payfixedLeg_[i]*((double)(payDates_[i+1] -
       valuationDate_ + 1)/
        (maturityDate_ - valuationDate_ + 1));

     sum = sum + notional_;
     duration = sum/val;
     cout << "fixed duration = " << duration << endl;

     return duration;
}
inline double calcModifiedDuration() {

   double val = calcDV01();
   double marketValue = getFixedValue();
   double MD = (val/(notional_ + marketValue))*10000;

   return MD;
}
inline void setFixedValue(double val) { value  = val; }
inline void setValuationDate(Date date) { valuationDate_ = date; }
inline double getFixedValue() {
   return value_;
}
inline void setNotional(double notional) { notional_ = notional; }
inline void setFrequency(int frequency) { payFrequency_ =
  frequency; }
inline void setFixedRate(double rate) { fixedLegRate_ = rate; }
inline void setMaturityDate(Date mat) { maturityDate_ = mat; }
std::vector<double> getPayFixed() { return payfixedLeg_; }
inline void calcPayFixed()
{
  std::vector<Date>::iterator iter;
  double val = 0;
  int diff = 0;
  Date dateDiff;
  int cnt = 0;

  for (iter = payDates_.begin(); iter != payDates_.end(); iter++)
  {
    if (payFrequency_ == 1)
    {
          if (cnt <= payDates_.size())
          {
                if (cnt + payFrequency_ < payDates_.size())
                   diff = payDates_[cnt+payFrequency_] -
                     payDates_[cnt]+1;
                else
                   diff = 0;

                if ((cnt != 0) && (cnt % payFrequency_ == 0))
                    val = notional_*(THIRTY/THREE_SIXTY)*
```

```
                        fixedLegRate_;
                 else
                     val = 0;
             }
             else
                val = notional_*(THIRTY/THREE_SIXTY)*fixedLegRate_;
       }
       else
       {
              if (cnt <= payDates_.size())
                  {
                if (cnt + payFrequency_ <= payDates_.size())
                    {
                  // subtract five because there are 5 less days in a
                  // 360 day year
                      diff = (payDates_[cnt+payFrequency_] -
                        payDates_[cnt] + 1) - 5;
                    }
                else
                  diff = 0;

                if ((cnt > 0) && ((cnt-1) % payFrequency_ == 0))
                  val = notional_*((double)diff/THREE_SIXTY)*
                    fixedLegRate_;
                else
                  val = 0;
                  }
                  else
                  {
                    val = 0;
                  }
       }
       cnt++;
       payfixedLeg_.push_back(val);
    } // for
  }
  private:
        std::vector<double> payfixedLeg_;
        double fixedLegRate_;
        double fixedLegBasis_;
        double duration_;
        double value_;
        double accrual_;
        double notional_;
        double basis_;
    int payFrequency_;
        Date effectiveDate_;
        Date maturityDate_;
        Date valuationDate_;
};

class Swap
{
  public:
        Swap() : notional_(NOTIONAL), maturity_("12/31/2010"),
          swapType(0) {}
        Swap(double notional, Date maturity, Date effectiveDate,
          Date settlementDate, Date valuation,
        std::map<double,double> liborRate, std::map<double,double> disc,
```

```
          double fixedRate, int type)
           : notional_(notional), maturity_(maturity),
             effectiveDate_(effectiveDate), settlementDate_(settlementDate),
               valuationDate_(valuation), floatRates_(liborRate),
               fixedRate_(fixedRate), swapType(type)
        {
          getNotional();
          calcPayDates(effectiveDate_,maturity_,valuationDate_);
          fixedLeg.setNotional(notional_);
          fixedLeg.setValuationDate(valuationDate_);
          fixedLeg.setFrequency(2);
          fixedLeg.setFixedRate(fixedRate);
          fixedLeg.setMaturityDate(maturity);
          fixedLeg.calcPayFixed();
          fixedLeg.setEffectiveDate(effectiveDate_);
          floatLeg.setMaturityDate(maturity);
          floatLeg.setFrequency(4);
          floatLeg.setNotional(notional_);
          floatLeg.setFloatRate(liborRate);
          floatLeg.setValuationDate(valuationDate_);
          setDiscountRates(disc);
          floatLeg.calcPayFloat();
          netPayments();
          calcDV01();
        }
        virtual ~Swap() {}
        inline double getNotional() {
                return notional_;
        }
        inline void setDiscountRates(std::map<double,double> rate) {
          discRates_ = rate;
        }
        inline double calcDV01() {

          double val = 0;

          if (swapType == 0)
                val = fixedLeg.calcDV01() - floatLeg.calcDV01();
          else
                val = floatLeg.calcDV01() - fixedLeg.calcDV01();

          cout << "Swap DV01 = " << val << endl << endl;

          return val;
        }
        inline void calcPayDates(Date tradeDate, Date endDate,
          Date valuation)
        {
          effectiveDate_ = tradeDate-1;

          if (effectiveDate_ == Date::SATURDAY)
                effectiveDate_ = effectiveDate_ + 2;
          else if (effectiveDate_ == Date::SUNDAY)
                effectiveDate_ = effectiveDate_ + 1;

          Date currDate = effectiveDate_;
            int cnt = 0;

          while (currDate <= endDate)
```

```
    {
            currDate.AddMonths(3);
            while (currDate.day > effectiveDate_.day)
        currDate = currDate - 1;

            if (currDate <= valuation)
            {
               if (currDate.day_of_week == Date::SATURDAY)
                      currDate = currDate + 2;
                   else if (currDate.day_of_week == Date::SUNDAY)
                      currDate = currDate + 1;
                   else if (currDate ==
                     currDate.ChristmasDay(currDate.year))
                      currDate = currDate + 1;

                   fixedAccruedDate_ = currDate;
            }
            if ((currDate <= endDate) && (currDate >= valuation))
            {
                   if (currDate.day_of_week == Date::SATURDAY)
                      currDate = currDate + 2;
                   else if (currDate.day_of_week == Date::SUNDAY)
                      currDate = currDate + 1;
                   else if (currDate ==
                     currDate.ChristmasDay(currDate.year))
                      currDate = currDate + 1;

                   payDates_.push_back(currDate);
                   cnt++;
             }
    }
  }
  inline void netPayments()
  {
    double val = 0.0;
    double y = 0.0;
    std::vector<double> fixed = fixedLeg.getPayFixed();
    std::vector<double> fl = floatLeg.getPayFloat();
    double x = 0;
    double sum = 0.0;
    double sumfix = 0.0;
    double sumfloat = 0.0;

    if (swapType == 0)
    {
      fixedAccrued_ =  notional_*fixedRate_*((valuationDate_ -
        fixedAccruedDate_))/THREE_SIXTY;
          floatAccrued_ = -notional_*floatRates_[0]*
            ((valuationDate_ - fixedAccruedDate_))/THREE_SIXTY_FIVE;
    }
    else
    {
          fixedAccrued_ = -notional_*fixedRate_*((valuationDate_ -
            fixedAccruedDate_))/THREE_SIXTY;
          floatAccrued_ =  notional_*floatRates_[0]*
            ((valuationDate_ - fixedAccruedDate_))/THREE_SIXTY_FIVE;
    }

    int cnt = 0;
    for (int i = 0; i < payDates .size(); i++)
```

```
            {
                x = (double) (payDates_[i+1] - payDates_[0] +
                  1)/THREE_SIXTY_FIVE;
                y = interpolate(discRates_[floor(x)],discRates_[ceil(x)],
                    floor(x),ceil(x)x;
                if (swapType == 0)
                {
                  val = (fixed[i] - fl[i])*y;
                  sumfix = sumfix + fixed[i]*y;
                  sumfloat = sumfloat - fl[i]*y;
                }
                else
                {
                  val = (fl[i] - fixed[i])*y;
                  sumfix = sumfix - fixed[i]*y;
                  sumfloat = sumfloat + fl[i]*y;
                }
                sum = sum + val;
            }
            fixedLeg.setFixedValue(sumfix);
            floatLeg.setFloatValue(sumfloat);

            cout << "Fixed Accrued = " << fixedAccrued_ << endl;
            cout << "Float Accrued = " << floatAccrued_ << endl;
            cout << "Accrued = " << fixedAccrued_ + floatAccrued_ << endl;
            cout << "Principal = " << sum << endl;
            cout << "Market Value = " << sum + (fixedAccrued_ + floatAccrued_)
              << endl;
        }
  private:
        int swapType;
        double notional_;
        double fixedAccrued_;
        double floatAccrued_;
        double fixedRate_;
        FloatingLeg floatLeg;
        FixedLeg fixedLeg;
        Date maturity_;
        Date fixedAccruedDate_;
        Date floatAccruedDate_;
        Date effectiveDate_;
        Date settlementDate_;
        Date valuationDate_;
        std::string index;
        std::map<double,double> discRates_;
        std::map<double,double> floatRates_;
        double value;
};

#endif _SWAP_H__
```

考虑一个具有如下特征的互换：

```
Notional = $3,300,000
Maturity = March 28, 2013
Effective Date = March 29, 2004
Valuation Date = October 14, 2005
Swap Rate = 3.969%
Floating Rate = 3 Mo. T-Bill
Basis Spread = 0.00
```

从 Bloomberg 所获取的一个文件中可知以下 3 个月美国短期国债数据（表 1.19）.

表 1.19

期限	3个月短期国债	贴现	期限	3个月短期国债	贴现
1DY	0.0400 000	0.999 708	9MO	0.044 518 8	0.967 342
2DY 0000	0.038 000 0	0.999 708	10MO	0.044 852 5	0.963 507
1WK	0.038 168 8	0.999 258	11MO	0.045 172 5	0.959 660
2WK	0.038 200 0	0.999 258	1YR	0.045 456 3	0.955 712
3WK	0.038 600 0	0.998 517	2YR	0.046 390 0	0.912 185
1MO	0.039 458 3	0.996 614	3YR	0.047 120 0	0.869 450
2MO	0.040 258 3	0.993 225	4YR	0.047 220 0	0.829 536
3MO	0.041 400 0	0.989 193	5YR	0.047 590 0	0.790 131
4MO	0.042 000 0	0.985 853	6YR	0.047 910 0	0.752 230
5MO	0.042 600 0	0.982 445	7YR	0.048 180 0	0.715 816
6MO	0.043 287 5	0.978 239	8YR	0.048 600 0	0.679 864
7MO	0.043 702 5	0.974 794	9YR	0.048 700 0	0.647 234
8MO	0.044 122 5	0.971 079	10YR	0.049 020 0	0.614 542

从所列的互换支付日期中计算现金流折现值，浮动（3个月期美国短期国债）利率与贴现率通过以下全局定义的函数来进行线性插值：

```
static double interpolate(double rate1, double rate2, double t1, double
t2, double x)
{
    double dy = rate2 - rate1;
    double dt = t2 - t1;
    double slope = dy/dt;

    return rate1 + slope*x;
}
```

利用 **floor** 与 **ceil** 的内置数学路径对需要插值的利率进行插值. 例如，在 `FloatLeg` 中的 `calcPayFloat` 函数，定义

```
floatVal = interpolate(floatLegRate_[floor(diff)],floatLegRate_[ceil(diff)],
  floor(diff),ceil(diff),diff);
```

特别需要注意的是，这仅是插值的一个例子. 在实践中，为了得到更加准确的插值数据，我们需要利用流动性较好的金融工具（如美国短期国债期货、欧洲美元期货）或者利用计量技术（如三次样条等）来自举得出收益率曲线.

主函数为：

MAIN.cpp

```
#include <strstrea.h>
#include <fstream.h>
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include <math.h>
#include <map>
#include "Swap.h"
#define SIZE_X 100

void main()
{
  cout.setf(ios::showpoint);
  cout.precision(8);
```

```
    cout << "Swap Pricing Pay Fixed   " << endl << endl;
    Date start = "3/29/2004";
    cout << "Start date = " << start << endl;
    Date maturity = "3/28/2013";
    cout << "Maturity = " << maturity << endl;
    Date valuation = "10/14/2005"; //today";
    cout << "Valuation = " << valuation << endl;
    Date effectiveDate = "today";
    double notional = 3300000;
    cout << "Notional = " << notional << endl;
    double swapRate = 0.03969;
    cout << "Swap Rate = " << swapRate << endl;

    std::vector<double> mat;
    std::map<double,double> libor;
    std::map<double,double> discRate;
    char buffer[SIZE_X];
    char dataBuffer[SIZE_X];
    char* str = NULL;
    double yr = 0.0;
    double rate = 0.0;
    int swapType = 1;  // receive fixed-pay float ; 1 = pay fixed-receive
                       // float

    const char* file = "c:\\swapData.txt";
    ifstream fin;              // input file stream
    fin.clear();
    fin.open(file);

    if (fin.good())
    {
        while (!fin.eof())
        {
            fin.getline(buffer,sizeof(buffer)/sizeof(buffer[0]));
            //cout << buffer << endl;
            istrstream str(buffer);
            // Get data
            str >> dataBuffer;
            yr = atof(dataBuffer);

            str >> dataBuffer;
            if (strcmp(dataBuffer,"MO") == 0)
                yr = (double) yr/12;
            else if (strcmp(dataBuffer,"WK") == 0)
                yr = (double) yr/52;
            else if (strcmp(dataBuffer,"DY") == 0)
                yr = (double) yr/365;
            mat.push_back(yr);

            str >> dataBuffer;
            rate = atof(dataBuffer);
            libor[yr] = rate;

            str >> dataBuffer;
            rate = atof(dataBuffer);
            discRate[yr] = rate;
        }
     }
     else
        cout << "File not good!" << "\n";

     fin.close();

     Swap s(notional,maturity,start,start+1,valuation,libor,discRate,
       swapRate,swapType);
}
```

得到以下结果：

```
Fixed Accrued = 5821.2000
Float Accrued = -5867.3096
Accrued = -46.109589
Principal = -176178.34
Market Value = -176224.45
```

风险度量如下：

```
fixed duration = 5.8713815
fixed DV01 = 1937.5559
float duration = 0.20821918
float DV01 = -68.712329
Swap DV01 = 1868.8436
```

1.13 在 Matlab 中为百慕大互换进行定价

利率互换具有期货合约的一些特征. 它们都是用以锁定未来的利率头寸，通常互换用于比交易所交易期货更长的期限. 财务经理如果想在取得对下降风险的保护的同时，在有利的情况放大收益，也可以通过持有互换期权来实现. 互换期权在到期时给持有人一个权利，但不是义务，来持有一个互换合约：举例来说，一个收到固定利率的互换头寸（或收到浮动利率的互换头寸），其中期权的执行价格设定为互换的固定部分利率. 百慕大互换期权，则是给持有人一个权利，在期权到期之前的各个预先设定的日期来持有互换合约. 以下通过 Matlab 来计算[23]百慕大互换期权：

bermudan_swaption.m

```
function []=bermudan_swaption()

%This program can work for arbitrary no. of factors. You have to specify no.
%of factors as well as volatility structure for each factor. The volatility
%structure can be obtained from principal component analysis of correlation
%matrix and adjusting to calibrated volatilities as done in excellent paper
%by Rebonato. See my web page for the references
%(http://www.geocities.com/anan2999). It does not take correlation
%structure as input. You can also specify CEV constant alpha for skew.
%Remember changing this constant changes effective volatility.

%randn('state',[1541045451;4027226640]) % add a good random number seed
%here if you wish.
%if you don't matlab will choose its own seed.

delta=.25; %Tenor spacing. usually .25 or .5

P=5000;  % No. of paths, do not try more than 5000 paths unless you are
         % very patient
T_e1=6.0;               %maturity of underlying swap in years(must be an
                        %exact multiple of delta)
T_x1=5.75;              %last exercise date of the swaption (must be an
                        %exact multiple of delta)
T_s1=3.0;               %lockout date (must be an exact multiple of delta)


T_e=T_e1/delta+1;
T_x=T_x1/delta+1;
T_s=T_s1/delta+1;
N=T_e;

F=2;    % number of factors. If you change this line also change volatility
        % structure appropriately
alpha=1.0;%CEV constant alpha for skew.Remember changing this value changes
          %effective volatility
```

```
%It is 1.0 for lognormal model.
k=.1; % strike, fixed coupon
pr_flag=+1; %payer receiver flag; assumes value of +1 for a payer swaption
%and a value of -1 for a receiver swaption.


n_spot=2;
L=repmat(.10,[P,T_e+1]);
vol=repmat(0,[T_e,F]);
for n=1:N,
   for f=1:F,
      if(f==1)
         vol(n,f)=.15; %volatility of first factor
      end
      if(f==2)
         vol(n,f)= (.15-(.009*(n)*.25).^.5); %volatility of second factor
      end
   end
end
%You can add more volatility factors in the above line but please also
%change F accordingly
%drift=repmat(0,[P,F]);
money_market=repmat(1,[T_x,P]);
swap=repmat(0,[T_x,P]);
B=repmat(1,[P,T_e]);


money_market(2,:)=money_market(1,:).*(1+delta*L(:,1))';
increment=repmat(0,[P,1]);
drift=repmat(0,[P,F]);

for t= 2 : T_x,

   t


  normal_matrix=randn([P,F]);
  drift(:,:)=0;
   for n= t : T_e,
      increment(:,1)=0;

              %      n
      for f=1:F,

         drift(:,f)=drift(:,f)+ delta*vol(n-n_spot+1,f).*
                      ((L(:,n).^alpha)./(1+delta.*L(:,n))); %

         increment(:,1)=increment(:,1)+vol(n-n_spot+1,f).*
                          (L(:,n).^alpha)./L(:,n)...
            .*(normal_matrix(:,f).*sqrt(delta)-.5.*vol(n-n_spot+1,f).*
             (L(:,n).^alpha)./L(:,n)...
            .*delta+drift(:,f).*delta);
      end

      L(:,n)=L(:,n).*exp(increment(:,1));
      L(L(:,n)<.00001,n)=.00001;
   end

   B(:,t)=1.0;
```

```
    for n=t+1:T_e,
       B(:,n)=B(:,n-1)./(1+delta.*L(:,n-1));
    end



    money_market(t+1,:)=money_market(t,:).*(1+delta*L(:,n_spot))';

    if((t>= T_s) & (t <=T_x))
       for n=t:(T_e-1), %//the swap leg is determined one date before
                        %//the end

                        swap(t,:)=swap(t,:)+  (B(:,n+1).*
                                       (L(:,n)-k).*pr_flag*delta)' ;
       end
    end
    n_spot=n_spot+1;

end



value=repmat(0,[P,1]);
stop_rule=repmat(T_x,[P,1]);

value(swap(T_x,:)>0,1) = (swap(T_x,swap(T_x,:)>0))';
coeff=repmat(0,[T_x,6]);

for t=(T_x-1):-1:T_s,
   i=0;
   a=0;
   y=0;
   for p=1:P,
      if (swap(t,p)> 0.0)
         i=i+1;
         a(i,1)=1;
         a(i,2)=swap(t,p);
         a(i,3)=swap(t,p)*swap(t,p);
         a(i,4)=money_market(t,p);
         a(i,5)=money_market(t,p)*money_market(t,p);
         a(i,6)=money_market(t,p)*swap(t,p);

         y(i,1)= money_market(t,p)/money_market(stop_rule(p,1),p) *
                    value(p,1);

      end

   end



   temp=inv(a'*a)*(a'*y);
   coeff(t,:)=temp';

   expec_cont_value=repmat(0,[P,1]);
   exer_value=repmat(0,[P,1]);

   expec_cont_value(:,1)=(coeff(t,1)+coeff(t,2).*swap(t,:)+
```

```
                                    coeff(t,3).*swap(t,:)...
      .*swap(t,:)+coeff(t,4).*money_market(t,:)+
        coeff(t,5).*money_market(t,:)...
      .*money_market(t,:)+coeff(t,6).*money_market(t,:).*swap(t,:))';

exer_value(swap(t,:)>0,1)=(swap(t,swap(t,:)>0))';

 value((exer_value(:,1)>expec_cont_value(:,1))&(swap(t,:)>0)',1)...
    =exer_value((exer_value(:,1)> expec_cont_value(:,1))&(swap(t,:)>0)',1);

 stop_rule((exer_value(:,1)>expec_cont_value(:,1))&(swap(t,:)>0)',1)=t;

end

   price=0;
   for p=1:P,
      price=price+ (value(p,1)/(money_market(stop_rule(p,1),p)))/P;

   end

   price
```

尾注

1. 这是一个典型的固定换浮动互换的情况.当对摊销本金型债券投资组合进行对冲，或需要对债券投资组合的全收益进行对冲时，就需要用到恒定到期互换（或指数摊销互换）与全收益互换.这些互换都没有相对应的期货品种.
2. 欧洲美元（Eurodollar）是指在美国境外的美国银行或者外国银行贷出的美元.欧洲美元利率是银行之间相互拆借的利率.
3. 显然，如果一个较长时间到期的期货合约的标的资产与利率高度正相关，那么期货价格将高于远期价格，因为期货每天都交割，所获得的盈利可以及时再投资于更高的利率水平上.类似地，当标的资产与利率高度负相关时，期货头寸将会带来直接损失，并且这个亏损将需要以更高的利率成本来融得.投资者此时会选择远期合约而不是期货合约，这样就不会受到利率浮动的影响.
4. Hull J(1997)，99.
5. Id. 99.
6. Id. 100.
7. Id. 100.
8. Id. 100.
9. Hull J(1997)，97.
10. 参见 http://www.academ.xu.edu/johnson/.
11. 实际上，投资经理只能购买 5 张合约，因为合约只能按整数来购买.因此，他将有 0.115 个单位的美国短期国债没有完全对冲，但如果他购买 6 张合约，则他又将有 0.985 个单位的美国短期国债过度对冲.

12. 可以用连续复利来计算久期 y_i，$1 \leqslant i \leqslant n$，

$$D = \sum_{i=1}^{n} t_i \left[\frac{c_i e^{-y_i t_i}}{B} \right]$$

其中 $B = \sum_{i=1}^{n} c_i e^{-y_i t_i}$.

13. “收益率修正久期”是指传统的分析公式中用恒定的贴现率来计算修正久期.
14. 计算 DV01，也就是将收益率曲线向上平移一个基点，用收益率加上 1 个基点来重新计算久期，然后再减去调整之前的久期.
15. 经允许，从 CBOT 利率互换组合白皮书“利用互换期货进行固定收益组合对冲”中重构.
16. Fabozzi F，Wiley(2002)，95 页 .《Term Structure，and Valuation Modeling》中 AudleyD、Chin R 和 Ramamuthy S 撰写的 “Term Structure Modeling”.
17. Id. 95.
18. 典型情况下，$n=2$，因为绝大多数债券采用半年复利.
19. “三次样条”是一种分段多项式函数，通过个别的多项式片断在一定的点（使用者指定）上拼接而成，这些点称为节点（knot point）. 三次样条函数是一个分段三次多项式方程，并在每个节点处二次可微. 在每个节点处，两边的斜率与曲率都相互契合. 指数立方函数将拟合通过各贴现点的指数曲线. 因此，三次样条和指数样条可用于在已知期限结构因子的情况下构建光滑的债券价格（收益率）曲线.

    见 James 与 Webber(2000)，《Interest Rate Modeling》，Wiley 出版社，430～432 页；Waggoner D(1997)；Pienaar R 与 Choudhry M 在 Fabozzi(2000)中的文章 “Fitting the Term Structure of Interest Rates Using the Cubic Spline Methodology”，157～185 页；O De la Grandville(2001)，《Bond Pricing and Portfolio Analysis》，MIT 出版社，248～252 页；Vasicek O 与 Fong H(1982)，“Term Structure Modeling Using Exponential Splines”，金融杂志 37 期，1982 年，339～361 页.
20. 由 Stefano Galiani（2003）开发.
21. 见《MATLAB Fixed-Income Toolkit User’ s Guide》，2002 年，MathWorks 出版社.
22. Date 类是由 James M. Curran(1994) 编写的开源库.
23. 由 Ahsan Amin 编写，详见 http://www. geocities. com/anan2999/.

第2章 copula函数

copula函数是为结构型信用产品定价的基本工具，它们拥有结合产品所含信用之间的相关依赖性的能力. copula函数可分为密度函数（椭圆型copulae）和母函数（阿基米德copulae）两类. 在为信贷违约掉期、信用互换指数（CDX）和抵押债务证券（CDO）等要求对参考实体的依赖性和相关结构进行建模的信用产品定价时，copulae十分有效. 在2.1节中，我们将给出copula函数的定义及基本性质. 在2.2节中，我们将讨论copula函数的分类，包括多元高斯copula和多元学生t copula. 在2.3节中，我们将回顾阿基米德copulae，它对于依赖性结构建模是一个大而灵活的copulae类. 在2.4节中，我们将讨论校准copulae；我们会回顾高斯和学生t copula的基于极大似然估计的误差配准（Exact Maximum Likelihood，EML）算法；还会讨论边际推理函数（Inference Functions for Margins，IFM）法和校准copula参数时使用的标准基于极大似然估计的误差配准算法；我们还会讨论Bouyè、Mashal和Zeevi方法；另外，我们还会给出以上两个校准方法的Matlab应用. 在2.5节中，我们将讨论并提供Galiani（2003）中所完成的校准真实市场数据的数值结论. 在2.6节中，我们会提供一些Excel copulae的例子.

2.1 copula函数的定义及基本性质

一个n维copula是具有如下性质的函数C：$[0,1]^n \to [0,1]$：

1. 对于任意一个$\boldsymbol{u}_k(k\in\{1,2,\cdots,n\})$，$C(\boldsymbol{u})$都是递增的.

2. 对于任意一个向量$\boldsymbol{u}\in[0,1]^n$，如果$\boldsymbol{u}$至少有一个坐标项为0，则$C(\boldsymbol{u})=0$；如果除去第k项坐标之外，$\boldsymbol{u}$的所有坐标项均为1，则$C(\boldsymbol{u})=\boldsymbol{u}_k$.

3. 给定顶点在定义域C中的超立方体$\boldsymbol{B}=[\boldsymbol{a},\boldsymbol{b}]=[a_1,b_1]\times[a_2,b_2]\times\cdots\times[a_n,b_n]$，对于$\boldsymbol{a}\leqslant\boldsymbol{b}$的每一个$\boldsymbol{a}$，$\boldsymbol{b}\in[0,1]^n$，超立方体的体积[1]$V_C(\boldsymbol{B})\geqslant 0$.

以上定义表示C是一个呈现均匀边际分布的多元分布函数. 一旦我们将copula函数的定义推广到向量或随机变量，上述性质的统计诠释就会更有意义. 但首先，我们需要引入一个辅助定理，它构成了copula研究方法的一个重要相关结论.

定理1（Sklar） *若G是边际分布为F_1，F_2，…，F_n的n维分布函数，则存在一个n维copula C，对于$\boldsymbol{x}\in\mathcal{R}^n$，我们有*

$$G(x_1,x_2,\cdots,x_n)=C(F_1(x_1),F_2(x_2),\cdots,F_n(x_n)) \tag{2.1}$$

此外，若F_1，F_2，…，F_n是连续的，则C是唯一的. 任何多元分布函数的多元边际（若是随机变量，则使用分布函数）和依赖性结构都是可以分开的，随后的copula函数进行了完整的描述. 通过copula函数的描述，Sklar定理表达了使用copula函数进行依赖性建模的基本思想. Scaillet（2000）指出，Sklar定理有一个重要的推论：

推论1 *若G和C分别为一个n维分布函数（连续单变量边际F_1，F_2，…，F_n）和一个n维copula函数，则对于任何$\boldsymbol{u}\in[0,1]^n$，我们有*

$$C(u_1,u_2,\cdots,u_n)=G(F_1^{-1}(u_1),F_2^{-1}(u_2),\cdots,F_n^{-1}(u_n)) \tag{2.2}$$

这里$F_i^{-1}(u_i)$代表累积分布函数的倒数，也就是说，对于$u_i\in[0,1]$，$F_i^{-1}(u_i)=\inf\{x:$

$F_i(x)\geqslant u_i\}$.

在之后几个小节中，我们会展示一个具体的 copula 所产生的模拟随机数的总体框架，从而明确式（2.2）的重要性. 令（X_1，X_2，…，X_n）′为一个随机变量的 n 维向量，这个随机变量的分布函数是（F_1，F_2，…，F_n），同时其联合分布函数是 G. 根据 Sklar 定理，若（F_1，F_2，…，F_n）为连续函数，则（X_1，X_2，…，X_n）′有唯一 copula，表达式如下：

$$G(x_1,x_2,\cdots,x_n) = P(X_1 \leqslant x_1,\cdots,X_n \leqslant x_n) = C(F_1(x_1),F_2(x_2),\cdots,F_n(x_n))$$

以上对 copula 函数的描述允许我们进一步重新建立以上定义 1 后面的两个性质. 实际上，性质(2)沿袭了所谓的“概率积分变换”（见 Casella and Berger[2000]）中所描述的事实，即若随机变量 X 和 Y 分别有连续分布函数 F_X 和 F_Y，则随机变量 $U=F_X(X)$ 和 $V=F_Y(Y)$ 在 [0，1] 中是均匀分布的，从而这是一个二元情况，

$$C(u,1) = P(U \leqslant u,V \leqslant 1) = P(U \leqslant u) = u$$

然后，

$$C(u,0) = P(U \leqslant u,V \leqslant 0) = 0$$

性质(3)保证了 copula 函数 C 遵循适当多元分布函数的定义性特征——为 $[0，1]^n$ 中的全部矩形子集加非负权.

通过应用 Sklar 定理及对分布函数和密度函数间关系的拓展[2]，我们能够轻松地推导出与 copula 函数 $C(F_1(x_1)，F_2(x_2)，\cdots，F_n(x_n))$ 相关的多元 copula 密度 $c(F_1(x_1)，F_2(x_2)，\cdots，F_n(x_n))$：

$$\begin{aligned} f(x_1,\cdots,x_n) &= \frac{\partial^n[C(F_1(x_1),F_2(x_2),\cdots,F_n(x_n))]}{\partial F_1(x_1)\cdots\partial F_n(x_n)} \cdot \prod_{i=1}^{n} f_i(x_i) \\ &= c(F_1(x_1),F_2(x_2),\cdots,F_n(x_n)) \cdot \prod_{i=1}^{n} f_i(x_i) \end{aligned}$$

这里我们可以定义

$$c(F_1(x_1),\cdots,F_n(x_n)) = \frac{f(x_1,\cdots,x_n)}{\prod_{i=1}^{n} f_i(x_i)} \tag{2.3}$$

如同我们将要在 2.2 节中见到的一样，为了校准其真实市场数据参数，相关的 copula 密度知识是非常重要的.

2.2 copula 函数的分类

copula 函数有很多种分类方法. 最常用的是高斯 copula 和属于椭圆分类的学生 t copula.[3]

2.2.1 多元高斯 copula

定义 1 令 $\boldsymbol{R}$ 为 $\mathrm{diag}(R)=1$ 的对称正定矩阵，Φ_R 为相关于矩阵 $\boldsymbol{R}$ 的标准多元正态分布，则我们定义这个多元高斯 copula 为

$$C(u_1,u_2,\cdots,u_n;\boldsymbol{R}) = \Phi_R(\Phi^{-1}(u_1),\Phi^{-1}(u_2),\cdots,\Phi^{-1}(u_n)) \tag{2.4}$$

其中 $\Phi^{-1}(u)$ 为正态累积分布函数的倒数. 相关的多元正态（multinormal）copula 密度可以从

等式（2.3）中得出：

$$c(\Phi(x_1),\cdots\Phi(x_n))=\frac{f^{\text{gaussian}}(x_1,\cdots,x_n)}{\prod_{i=1}^{n} f_i^{\text{gaussian}}(x_i)}=\frac{\frac{1}{(2\pi)^{n/2}|\boldsymbol{R}|^{1/2}}\exp\left(-\frac{1}{2}x'\boldsymbol{R}^{-1}x\right)}{\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x_i^2\right)}$$

从而固定 $u_i=\Phi(x_i)$，用 $\zeta=(\Phi^{-1}(u_1),\ \cdots,\ \Phi^{-1}(u_n))'$表示高斯单变量逆分布函数的向量，我们得出下式：

$$c(u_1,u_2,\cdots,u_n;\boldsymbol{R})=\frac{1}{|\boldsymbol{R}|^{1/2}}\exp\left[-\frac{1}{2}\zeta'(\boldsymbol{R}^{-1}-I)\zeta\right] \tag{2.5}$$

图 2.1 表示出式（2.5）中所描述的是对于相关量 r 的二元高斯 copula 密度曲面.

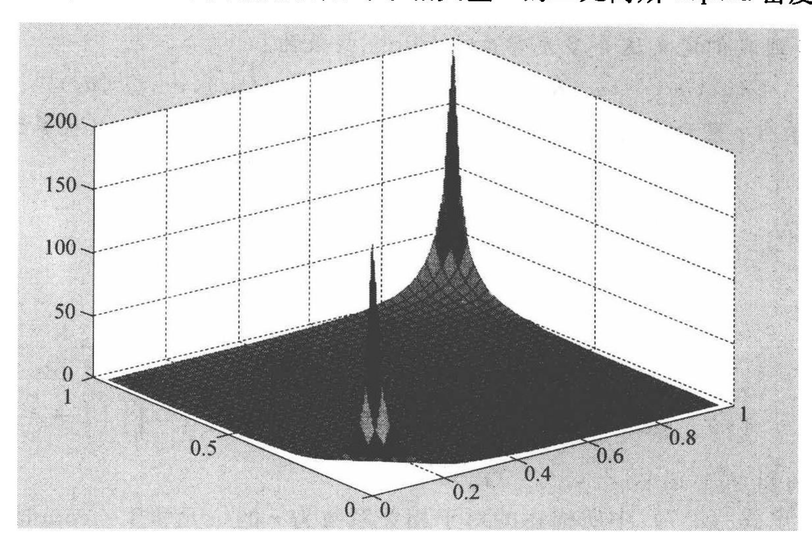


图 2.1　二元高斯 copula 密度

资料来源：Galiani S(2003).

以下 Matlab 编码是用来计算高斯 copula 密度曲面：

gaussian_copula_density.m

```
% This script plot the density of a bivariate gaussian copula function
% Pairwise correlation is set at 50%

R=ones(2,2);
r=.5;

R(1,2)=r;
R(2,1)=r;
X=zeros(2,1);
U=zeros(2,1);
gc=zeros(39,39);
h=0;

for i=0.025:.025:.975
    h=h+1;
    k=0;
    for j=0.025:.025:.975
```

```
        X=[i;j];
        k=k+1;
        U=norminv(X);
        block1=1/(det(R)^0.5);
        block2=-0.5*U'*(inv(R)-ones(2,2))*U;
        gauss_grid(h,k)=block1*exp(block2);
    end
end
surf(gauss_grid)
```

2.2.2 多元学生 t copula

定义 2 令 $\boldsymbol{R}$ 为 $\mathrm{diag}(R)=1$ 的对称正定矩阵，$T_{\boldsymbol{R},v}$ 为相关于矩阵 $\boldsymbol{R}$ 的自由度为 v 的标准多元学生 t 分布[4]，则我们定义这个多元学生 t copula 函数为：

$$C(u_1,u_2,\cdots,u_n;\boldsymbol{R},v) = T_{\boldsymbol{R},v}(t_v^{-1}(u_1),t_v^{-1}(u_2),\cdots,t_v^{-1}(u_n)) \tag{2.6}$$

其中 $t_v^{-1}(u)$ 为学生 t 累积分布函数的倒数. 相关的学生 t copula 密度可以从等式（2.3）中得出如下：

$$\begin{aligned} c(u_1,u_2,\cdots,u_n;\boldsymbol{R},v) &= \frac{f^{\text{Student}}(x_1,\cdots,x_n)}{\prod_{i=1}^{n} f_i^{\text{Student}}(x_i)} \\ &= |\boldsymbol{R}|^{-1/2}\frac{\Gamma\left(\frac{v+n}{2}\right)}{\Gamma\left(\frac{v+1}{2}\right)}\left[\frac{\Gamma\left(\frac{v}{2}\right)}{\Gamma\left(\frac{v+1}{2}\right)}\right]^n \frac{\left(1+\frac{\zeta'\boldsymbol{R}^{-1}\zeta}{v}\right)^{-\frac{v+n}{2}}}{\prod_{i=1}^{n}\left(1+\frac{\zeta_i^2}{v}\right)^{-\frac{v+1}{2}}} \end{aligned} \tag{2.7}$$

其中 $\zeta=(t_v^{-1}(u_1),\ t_v^{-1}(u_2),\ \cdots,\ t_v^{-1}(u_n))'$.

图 2.2 表示出式（2.7）中所描述的对于相关系数为 r 的二元学生 t copula 密度曲面.

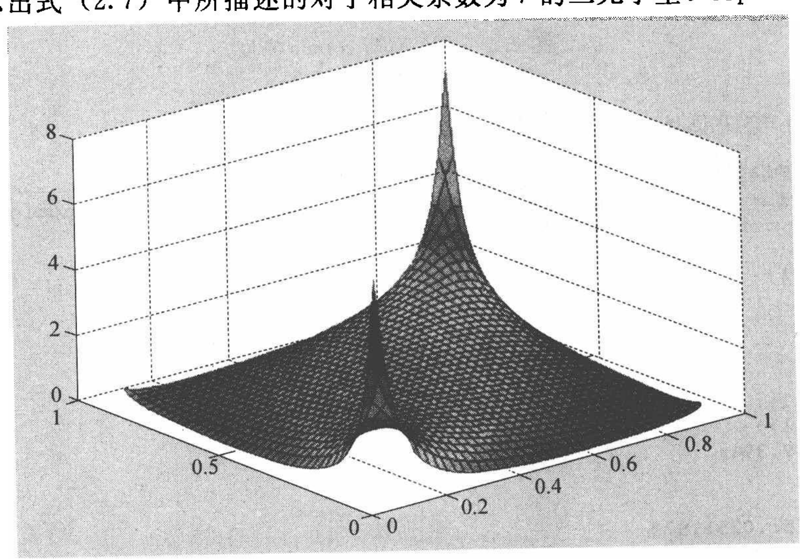


图 2.2 二元学生 t copula 密度

资料来源：Galiani S(2003).

以下 Matlab 编码是用来计算学生 t copula 密度曲面：

t_copula_density1.m

```
% This script plots che bivariate Student's t copula with 50% correlation
% and 3 degrees of freedom

R=ones(2,2);
r= .5;

R(1,2)=r;
R(2,1)=r;
X=zeros(2,1);
U=zeros(2,1);
gc2=zeros(39,39);
h=0;
DoF=3; % degrees of freedom
d=2; %dimension
Block_1=0;
Block_2=0;
for i=0.025:.025:.975
    h=h+1;
    k=0;
    for j=0.025:.025:.975
        X=[i;j];
        k=k+1;
        y=(tinv(X,DoF));
        A=gamma((DoF+d)/2)*((gamma(DoF/2))^(d-1));
        B=((gamma((DoF+1)/2))^d)*((det(R))^0.5);
        Block_1=A/B;
        A=0;
        B=0;

        Block_2=1;
        for l=1:d
            C=(1+((y(l)^2)/DoF))^(-(DoF+1)/2);
            Block_2=Block_2*C;
            C=0;
        end
        D=y'*inv(R)*y/DoF;
        Block_3=(1+D)^(-(DoF+d)/2);
        D=0;
        t_grid(h,k)=(Block_1/Block_2)*Block_3;
    end
end

surf(t_grid)
```

2.3 阿基米德 copulae

阿基米德 copulae 构成了 copula 函数的一个重要类别，不仅仅是因为其具有分析回溯性（很多常规的阿基米德 copulae 具有封闭型表达式），还因为它可以提供很多不同的依赖性结构.

下面是 Nelson 在 1999 年做的分析. 请考虑函数 φ：$[0,1]\to[0,\infty)$，满足：

- φ 是连续的.
- 对于所有 $u\in[0,1]$，$\varphi'(u)<0$.

- $\varphi(1)=0$.

定义 φ 的拟逆为函数 $\varphi^{[-1]}$：$[0,\infty)\to[0,1]$. 从而

$$\varphi^{[-1]}(t)=\begin{cases}\varphi^{-1}(t), & 0\leqslant t\leqslant\varphi(0)\\ 0, & \varphi(0)\leqslant t\leqslant\infty\end{cases}$$

现在，如果 φ 是凸性的，则定义函数 C: $[0,1]^2\to[0,1]$ 为

$$C(u,v)=\varphi^{[-1]}[\varphi(u)+\varphi(v)] \tag{2.8}$$

它是一个阿基米德 copula，并且 φ 称为这个 copula 的产生者. 此外，如果 $\varphi(0)=\infty$，则 φ 的拟逆描述了一个普通的反函数（即 $\varphi^{[-1]}=\varphi^{-1}$），并且我们称 φ 为一个严格产生者，称 C 为一个严格阿基米德 copula.

Gumbel copula 当 $\theta\geqslant 1$ 时，令 $\varphi(t)=(-\ln t)^\theta$，则通过使用等式（2.8），我们得到

$$C_\theta^{\text{Gumbel}}(u,v)=\varphi^{-1}[\varphi(u)+\varphi(v)]=\exp\{-[(-\ln u)^\theta+(-\ln v)^\theta]^{1/\theta}\}$$

Clayton copula 当 $\theta\in[-1,\infty)\setminus\{0\}$时，令 $\varphi(t)=(t^{-\theta}-1)/\theta$，则通过使用等式（2.8），我们得到

$$C_\theta^{\text{Clayton}}(u,v)=\max[(u^{-\theta}+v^{-\theta}-1)^{1/\theta},0]$$

注意到，如果 $\theta>0$，则 $\varphi(0)=\infty$，从而可以化简上一等式为

$$C_\theta^{\text{Clayton}}(u,v)=(u^{-\theta}+v^{-\theta}-1)^{1/\theta} \tag{2.9}$$

Frank copula 令 $\varphi(t)=-\ln\dfrac{e^{-\theta t}-1}{e^{-\theta}-1}$，$\theta\in\mathcal{R}\setminus\{0\}$，则通过使用等式（2.8），我们得到

$$C_\theta^{\text{Frank}}(u,v)=-\frac{1}{\theta}\ln\left[1+\frac{(e^{-\theta u}-1)(e^{-\theta v}-1)}{e-\theta-1}\right]. \tag{2.10}$$

我们可以把阿基米德 copulae 的架构推广到多元情况. 按照 Embrechts、Lindskog 和 McNeil（2001）的分析，我们可以得出以下定理：

定理 2（Kimberling） *令 φ: $[0,1]\to[0,\infty)$ 是一个连续的严格递减函数，并且 $\varphi(0)=\infty$，$\varphi(1)=0$，再令 φ^{-1}是 φ 的反函数，则对于所有 $n\geqslant 2$，函数 C: $[0,1]^n\to[0,1]$ 定义为*

$$C(u_1,u_2,\cdots,u_n)=\varphi^{-1}[\varphi(u_1)+\varphi(u_2)+\cdots+\varphi(u_n)]$$

它是一个 n 维阿基米德函数当且仅当 φ^{-1} 在区间 $[0,\infty)$ 上完全单调[5].

2.4 校准 copulae

对于真实市场的数据，校准 copulae 参数是结构型信用产品定价的一个重要步骤，它可以确保定价的精确性和稳健性. 在下面的分析中我们考虑一个由时间序列 $\boldsymbol{X}=(X_{1t},X_{2t},\cdots,X_{Nt})_{t=1}^T$表示的随机抽样，其中 N 代表相关资产的数量（比如贷款），T 代表可用观测（以日、月或年为单位）的数量.

2.4.1 基于精确极大似然估计的误差配准算法（EML）

令 Θ 为一个参数空间，而 θ 为待预测参数的 k 维向量. 令 $L_t(\theta)$ 和 $l_t(\theta)$ 分别为在 t 时间观测的似然函数（likelihood function）和对数似然函数（log-likelihood function）. 定义对数似然函数 $l(\theta)$ 为

$$l(\theta)=\sum_{t=1}^{T}l_t(\theta) \tag{2.11}$$

考虑到由等式（2.3）所表示的密度函数的正则表达式，我们可以将等式（2.11）推广如下：

$$l(\theta)=\sum_{t=1}^{T}\ln c(F_1(x_1^t),\cdots,F_N(x_N^t))+\sum_{t=1}^{T}\sum_{n=1}^{N}\ln f_n(x_n^t) \tag{2.12}$$

定义极大似然估计值为向量$\hat{\theta}$，满足

$$\hat{\theta}=(\hat{\theta}_1,\hat{\theta}_2,\cdots,\hat{\theta}_k)\in \arg\max\{l(\theta):\theta\in\Theta\}$$

高斯 copula 令 $\Theta=\{R:\ R\in\mathcal{R}^{N\times N}\}$ 表示以 R 为对称正定矩阵的参数空间. 对等式（2.5）中给出的高斯 copula 密度使用等式（2.12），我们得出

$$l^{\text{gaussian}}(\theta)=-\frac{T}{2}\ln|R|-\frac{1}{2}\sum_{t=1}^{T}\zeta_t'(R^{-1}-I)\zeta_t \tag{2.13}$$

假设等式（2.13）中的对数似然函数关于 θ 是可微的，并且定义等式 $\frac{\partial}{\partial\theta}\ell(\theta)=0$ 的结果为全局极大值，我们很容易发现，由等式（2.13）给出对数似然的高斯 copula 的极大似然估计值$\hat{\theta}=\hat{R}$：

$$\frac{\partial}{\partial R^{-1}}\ell^{\text{gaussian}}(\theta)=\frac{T}{2}R-\frac{1}{2}\sum_{t=1}^{T}\zeta_t'\zeta_t$$

因此，

$$\hat{R}=\frac{1}{T}\sum_{t=1}^{T}\zeta_t'\zeta_t \tag{2.14}$$

学生 *t* copula 令 $\theta=\{(v,\ R):\ v\in[2,\ \infty),\ R\in\mathcal{R}^{N\times N}\}$ 表示以 R 为对称正定矩阵的参数空间. 我们可以对等式（2.7）中给出的学生 t copula 密度使用等式（2.14）. 在这种情况下，计算变得更加复杂，我们得出下式：

$$\ell^{\text{Student}}(\theta)=T\ln\frac{\Gamma\left(\frac{v+N}{2}\right)}{\Gamma\left(\frac{v}{2}\right)}-NT\ln\frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)}-\frac{T}{2}\ln|R|-\frac{v+N}{2}\sum_{t=1}^{T}\ln\left(1+\frac{\zeta_t'R^{-1}\zeta_t}{v}\right)+\frac{v+1}{2}\sum_{t=1}^{T}\sum_{n=1}^{N}\ln\left(1+\frac{\zeta_{nt}^2}{v}\right) \tag{2.15}$$

与高斯 copula 的情况不同，通过 EML 方法进行的学生 t copula 的校准更加复杂，因为那要求一个对边际参数和与依赖性结构相关的参数的同时估计（见 Johnson 和 Kotz（1972））. 但是在这个过程中，就如 Mashal 和 Naldi（2002）所指出的，需要大量的数据支持和繁多的计算. 因此，一般采用边际推断函数方法（IFM）和正则极大似然方法（CML）.

2.4.2 边际推断函数方法（IFM）

基于 Joe 和 Xu（1996）的研究，IFM 利用 copula 理论的基本思想（即单边际与依赖性结构的分离），将等式（2.12）表示为

$$\ell(\theta)=\sum_{t=1}^{T}\ln c(F_1(x_1^t;\theta_1),\cdots,F_N(x_N^t;\theta_N);\boldsymbol{\alpha})+\sum_{t=1}^{T}\sum_{n=1}^{N}\ln f_n(x_n^t;\theta_n) \tag{2.16}$$

等式（2.16）的特性依靠单变量边际 $\boldsymbol{\theta}=(\theta_1, \cdots, \theta_N)$ 参数的向量和 copula 参数向量 $\boldsymbol{\alpha}$ 的分离. 换句话说，copula 市场数据参数的校准是通过一个两阶段程序来执行的：

1. 通过 EML 方法对边际单变量 $\boldsymbol{\theta}=(\theta_1, \cdots, \theta_N)$ 参数向量进行估计. 例如，考虑第 i 个相关资产的时间序列，我们可以得出[6]：

$$\hat{\theta}_i = \arg\max_{\theta_i} \sum_{t=1}^{T} \ln f_i(x_i^t; \theta_i)$$

2. 使用前一个估计值$\hat{\boldsymbol{\theta}}=(\hat{\theta}_1, \cdots, \hat{\theta}_N)$ 来对 copula 参数向量 $\boldsymbol{\alpha}$ 进行估计：

$$\hat{\alpha}_{\mathrm{IFM}} = \arg\max_{\alpha} \sum_{t=1}^{T} \ln c(F_1(x_1^t; \hat{\theta}_1), \cdots, F_N(x_n^t; \hat{\theta}_N); \boldsymbol{\alpha})$$

以上的 IFM 估计值定义为向量 $\boldsymbol{\theta}^{\mathrm{IFM}}=(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}}_{\mathrm{IFM}})$.

2.4.3 正则极大似然方法（CML）

EML 和 IFM 方法都基于外生条件，所以是强加在单变量边际参数形式上的[7]. 另外一种方法是在边际的分布形势上不要求任何先验假设的 CML 方法，它依赖于实证边际变换（empirical marginal transformation）的概念. 实证边际变换趋向于用经验分布函数$\hat{F}_n(\cdot)$未知参数的边际$\hat{F}_n(\cdot)$，其中 $n=1, \cdots, N$，定义如下：

$$\hat{F}_n(\cdot) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{X_{nt} \leqslant \cdot\}} \quad , \quad n = 1, \cdots, N, \tag{2.17}$$

其中 $1_{\{X_{nt} \leqslant \cdot\}}$表示指标函数. CML 则是通过两步来实施的：

1. 使用经验边际分布来进行从初始数据集合 $X=(X_{1t}, X_{2t}, \cdots, X_{Nt})_{t=1}^{T}$ 到统一形式变量（uniform variates）的变换. 即对于 $t=1, \cdots, T$，令

$$\hat{u}_t = (\hat{u}_1^t, \hat{u}_2^t, \cdots, \hat{u}_N^t) = \lfloor \hat{F}_1(X_{1t}), \hat{F}_2(X_{2t}), \cdots, \hat{F}_N(X_{Nt}) \rfloor$$

2. 通过下列相关性对 copula 参数向量 $\boldsymbol{\alpha}$ 进行估计：

$$\hat{\boldsymbol{\alpha}}_{\mathrm{CML}} = \arg\max_{\boldsymbol{\alpha}} \sum_{t=1}^{T} \ln c(\hat{u}_1^t, \hat{u}_2^t, \cdots, \hat{u}_N^t; \boldsymbol{\alpha})$$

以上的 CML 估计值定义为向量 $\boldsymbol{\theta}^{\mathrm{CML}}=\hat{\boldsymbol{\alpha}}_{\mathrm{CML}}$.

2.5 校准真实市场数据的数值结论

在这一节，我们将会展示一个学生 t copula 的参数对真实市场数据的校准所使用的 CML 方法的应用（见 Galianil（2003））. 考虑一个由 4 只股票（Fiat、Merrill Lynch、Ericsson 和 British Airways）组成的投资组合，我们有从 1999 年 8 月 14 日到 2003 年 7 月 15 日之间的 990 个日观测数据. 这里我们提供两种方法：第一种是由 Bouyè 等人开发，其基于一个相关矩阵的递归优化过程；第二种是由 Mashal 和 Zeevi 提出，其基于 Kendall's tau 所给出的秩相关估计值.

2.5.1 Bouyè、Durrelman、Nikeghbali、Riboulet 和 Roncalli 方法

这一过程是由一系列后续步骤组成的，其可被概括并实行如下：

1. 从股票收益的随机抽样 X 开始，使用在 2.4.3 节中给出的经验边际变换和正则极大似然方法（CML）将初始数据集合转换成为统一变量集合$\hat{U}$.

2. 对于在一个指定值域中自由度为 v 的任何值，使用下列步骤对相关矩阵 $\boldsymbol{R}_v^{\text{CML}}$ 进行估计.

ⅰ. 对于所有限定日期 t，使 $\zeta_t=(t_v^{-1}(\hat{u}_{1t}),\ t_v^{-1}(\hat{u}_{2t}),\ \cdots,\ t_v^{-1}(\hat{u}_{2t}))'$，其中 $t=1,\ \cdots,\ T$.

ⅱ. 使用等式（2.14）来估计高斯 copula 的相关矩阵的精确极大似然估计值$\hat{R}$，则 $R_0=\hat{R}$.

ⅲ. 通过下列递归来得到 $R_{v,k+1}$：

$$R_{v,k+1}=\frac{v+N}{Tv}\sum_{t=1}^{T}\frac{\xi_t'\xi_t}{1+\frac{\xi_t'R_{v,k}^{-1}\xi_t}{v}}$$

ⅳ. 重新调整矩阵的元，从而得到单位对角线元素：

$$(R_{v,k+1})_{i,j}=\frac{(R_{v,k+1})_{i,j}}{\sqrt{(R_{v,k+1})_{i,i}(R_{v,k+1})_{j,j}}}$$

ⅴ. 重复步骤 ⅲ ～ ⅳ，直到 $R_{v,k+1}=R_{v,k}$ 而且 $R_v^{\text{CML}}=R_{v,k}$.

3. 通过最大化学生 t copula 密度的对数似然函数，我们可以找到自由度的 CML 估计值 v^{CML}：

$$v^{\text{CML}}=\arg\max_{v\in\Theta}\sum_{t=1}^{T}\log c^{\text{Student}}(\hat{u}_1^t,\hat{u}_2^t,\cdots,\hat{u}_N^t:R_v^{\text{CML}},v)$$

图 2.3 是 t copula 密度的对数似然函数相对于自由度函数的描点图，从中可以看出，所估计的自由度数是 10.

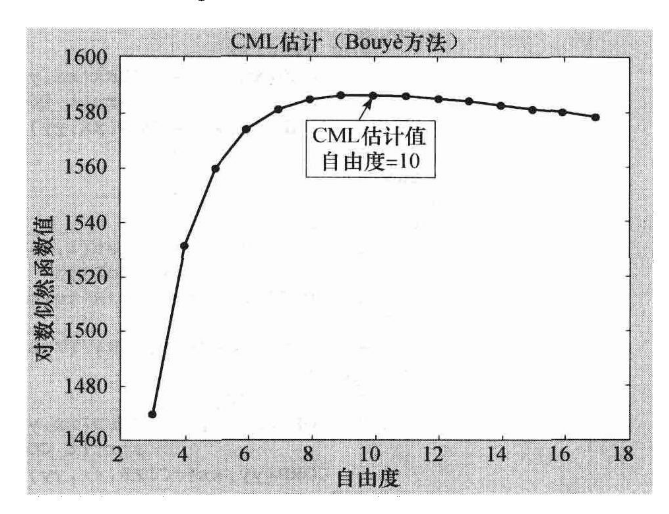


图 2.3 使用 Bouyè 方法计算的 t copula 密度的对数似然函数

资料来源：Galiani S(2003).

IMF_t_CORR.m

```
function [U_sample,CORR]=IFM_t_CORR(R,DoF,M)

% This function, starting from a random sample M, estimate the correlation
% matrix for the Student's t copula via the Bouye(2000), p.42 algorithm.
% Moreover it returns the uniform variate sample via the
% probability-integral transformation with Student's t margins.
% R:    correlation matrix for the gaussian copula
% DoF:  degrees of freedom for the multivariate t copula density
% M:    random sample of equity returns

N=size(R,2);
T=size(M,1);
U_emp=zeros(size(M));
U_st=zeros(size(M));
t_CORR=zeros(N);
CORR=zeros(N);

for n=1:N
    U_emp(:,n)=emp_dis(M(:,n));
end
```

```
for i=1:T
    U_st(i,:)=tinv(U_emp(i,:),DoF);
end
for k=1:100
    dummy_CORR=zeros(N);

        if k==1
            for i=1:T
                term=(U_st(i,:)'*U_st(i,:))/(1+(U_st(i,:)*
                        inv(R)*U_st(i,:)'/DoF));
                dummy_CORR=dummy_CORR+term;
            end
            t_CORR=dummy_CORR*(DoF+N)/(T*DoF);
            for xx=1:N
                for yy=1:xx
                   CORR(xx,yy)=t_CORR(xx,yy)/(sqrt(t_CORR(xx,xx))*
                                    sqrt(t_CORR(yy,yy)));
                   CORR(yy,xx)=CORR(xx,yy);
                end
            end
        else
            for i=1:T
                term=(U_st(i,:)'*U_st(i,:))/(1+(U_st(i,:)*inv(CORR)*
                        U_st(i,:)'/DoF));
                dummy_CORR=dummy_CORR+term;
            end
            t_CORR=dummy_CORR*(DoF+N)/(T*DoF);
            for xx=1:N
                for yy=1:xx
                   CORR(xx,yy)=t_CORR(xx,yy)/(sqrt(t_CORR(xx,xx))*
                                    sqrt(t_CORR(yy,yy)));
                   CORR(yy,xx)=CORR(xx,yy);
                end
            end
        end
end

U_sample=U_st;
```

emp_dis1.m

```
function X=emp_dis(Y)
% Compute the empirical marginal transformation, for the asset return
% samples, as described in Mashal(2002), "Beyond Correlation", pag.15
% As input we provide the i-th row of our d-columns sample
% i: #observation         d: #asset in the basket

X=zeros(length(Y),1);                    % create a vector of d dimension
count=0;
for k=1:length(Y)
    for s=1:length(Y)
        if Y(s)<=Y(k)
            count=count+1;
        end
    end
    X(k)=count/length(Y);                % compute the empirical
                                         % distribution function
    if X(k)==1
```

```
        X(k)=.999999;                         % otherwise, if 1, then we have
                                              % problem in using the t inverse
    end
    count=0;
end
```

t_copula_density2.m

```
function c=t_copula_density(U,DoF,Corr)

% gives the t-copula density for fixed input of "U" ( t-th row of the
% uniform sample with N elements), degrees of freedom "DoF", and
% correlation matrix "Corr"
%
N=length(U);
A=0;
B=0;
C=0;
D=0;
Block_1=0;
Block_2=1;
Block_3=0;

% Now, we split formula 4, pag. 12 of Mashal(2002) "Beyond Correlation" in
% 3 blocks

A=gamma((DoF+N)/2)*((gamma(DoF/2))^(N-1));
B=((gamma((DoF+1)/2))^N)*((det(Corr))^0.5);
Block_1=A/B;

for n=1:N
    C=(1+((U(n)^2)/DoF))^(-(DoF+1)/2);
    Block_2=Block_2*C;
end

D=U'*inv(Corr)*U/DoF;
Block_3=(1+D)^(-(DoF+N)/2);

c=(Block_1/Block_2)*Block_3;
```

pseudo_sample_IFM.m

```
% Starting from a random sample M, this script, generates a matrix (T*N) of
% uniform variates through the probability-integral transformation with
% gaussian margins.

% T:    # of observation in the time series for each asset
% N:    # of assets

M = xlsread('C:\copula\DataSet_new.xls'); % equity return database
N=size(M,2);
T=size(M,1);
ML_CORR=zeros(N);
Average=zeros(N,1);
Deviation=zeros(N,1);
U_nor1=zeros(size(M));
U_nor2=zeros(size(M));

for i=1:N
    Average(i)=mean(M(:,i));
```

```
    Deviation(i)=std(M(:,i),1);
    U_nor1(:,i)=normcdf((M(:,i)-Average(i))/Deviation(i));

end

for i=1:T
    U_nor2(i,:)=norminv(U_nor1(i,:));
    ML_CORR=ML_CORR+U_nor2(i,:)'*U_nor2(i,:);
end

ML_CORR=ML_CORR/T;
```

t_copula_CML

```
function MLE=t_copula_CML(DoF)
% calculate the target function to be maximized as described in Step 3
% pag. 43 of Mashal's (2002) paper
pseudo_sample
T=size(U,1);                    % number of observation for each stock
Log_L=zeros(T,1);
Sum_Log_L=0;
% For each observation compute the t-copula density and sum over
for t=1:T
    Log_L(t)=log(t_copula_density(U(t,:),DoF,CORR_EST));
    Sum_Log_L=Sum_Log_L+Log_L(t);
end

MLE=Sum_Log_L;
```

2.5.2 Mashal 和 Zeevi 方法

就像 Mashal 和 Zeevi（2002）指出的，受到因接近奇异矩阵的转置（inversion of close to singular matricè）而导致的数值不稳定性影响，在处理较多相关资产和大量数据集合时，Bouyè 方法的计算量很大. 因此，Mashal 和 Zeevi 提出应通过秩相关估计值（或称为 Kendal l’s tau）来估计学生 t copula 的相关矩阵. 得出的结论包含在下面的定理中.

定理 3 令 $X\sim E_N(\mu,\ \Sigma,\ \varphi)$，对于 $i,\ j\in\{1,\ 2,\ \cdots,\ N\}$，$X_i$ 和 X_j 是连续的，则

$$\tau(X_i,X_j)=\frac{2}{\pi}\arcsin R_{i,j} \tag{2.18}$$

其中 $E_N(\mu,\ \Sigma,\ \varphi)$ 表示带有参数（μ，Σ，φ）的 N 维椭圆分布，$\tau(X_i,\ X_j)$ 和 $R_{i,j}$ 分别表明对于随机变量（X_i，X_j）的 Kendall’s tau[9] 和 Pearson 线性相关系数.

证明 见 Lindkog、McNeil 和 Schmock（2001）.

1. 从股票价格的随机抽样 X 开始，使用在 2.4.3 节中给出的经验边际变换将初始数据集合转换成统一变量集合 $\hat{U}$.

2. 从等式（2.18）中估计相关矩阵 $\boldsymbol{R}^{\mathrm{CML}}$.

3. 通过最大化学生 t copula 密度的对数似然函数，我们可以找到自由度的 CML 估计值 v^{CML}：

$$v^{\text{CML}} = \arg\max_{v\in\Theta}\sum_{t=1}^{T}\log c^{\text{Student}}(\hat{u}_1^t,\hat{u}_2^t,\cdots,\hat{u}_N^t:R^{\text{CML}},v)$$

图 2.4 是 t copula 密度的对数似然函数相对于自由度函数的描点图，从中可以看出，所估计的自由度数是 9.

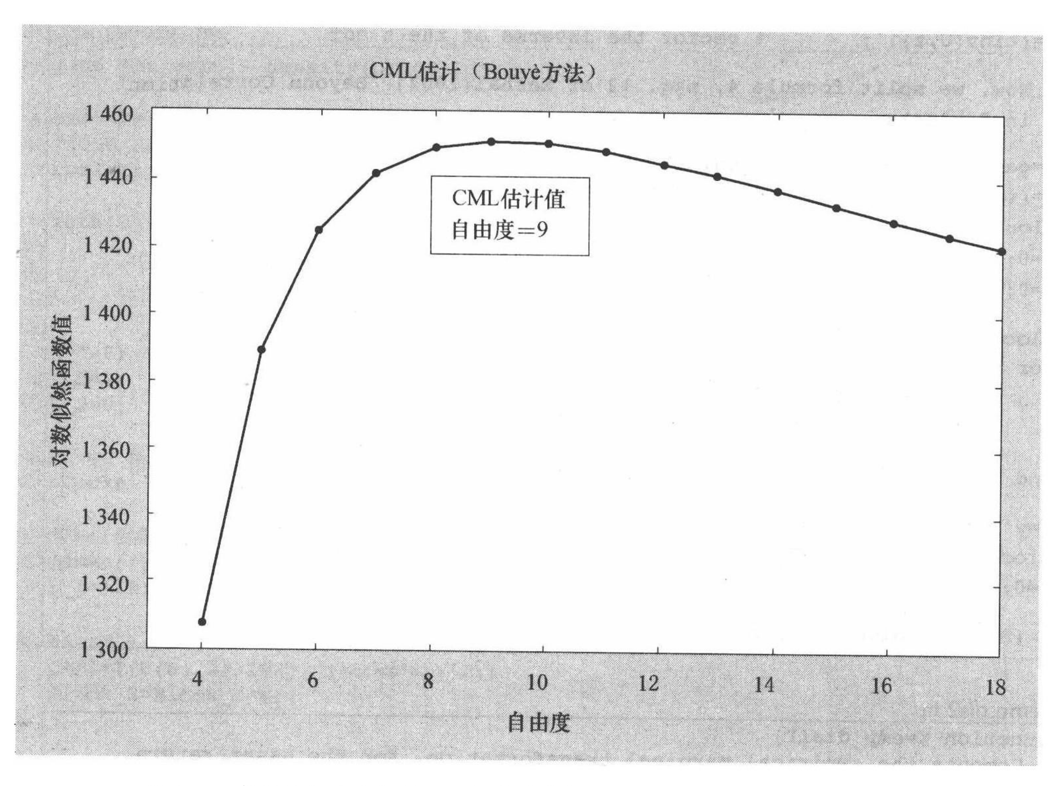


图 2.4　使用 Mashal 和 Zeevi 方法计算的 t copula 密度的对数似然函数

资料来源：Galiani S(2003).

表 2.1 展示了对应的校准相关矩阵 $\boldsymbol{R}_9^{\text{CML}}$.

表 2.1　相关系数矩阵

1	0.34 771	0.814 75	0.776 31
0.34 771	1	0.626 66	0.657 06
0.814 75	0.626 66	1	0.772 88
0.776 31	0.657 06	0.7728 8	1

以下是 Mashal 和 Zeevi 方法的 Matlab 应用：

t_copula_density3.m

```
function c=t_copula_density(U,DoF,Corr)

% gives the t-copula density for fixed input of "U" ( i-th row of the
% pseudo sample with d elements), degrees of freedom "DoF", and
% correlation matrix "Corr"
```

```
d=length(U);
y=zeros(d,1);          % create the vector of univariate t r.v.s.
z=zeros(d,1);
z=DoF;
y=(tinv(U,z))';        % vector the inverse of the t cdf

% Now, we split formula 4, pag. 12 of Mashal(2002) "Beyond Correlation"
% in 3 blocks

A=gamma((DoF+d)/2)*((gamma(DoF/2))^(d-1));
B=((gamma((DoF+1)/2))^d)*((det(Corr))^0.5);
Block_1=A/B;
A=0;
B=0;

Block_2=1;
for k=1:d
    C=(1+((y(k)^2)/DoF))^(-(DoF+1)/2);
    Block_2=Block_2*C;
    C=0;
end

D=y'*inv(Corr)*y/DoF;
Block_3=(1+D)^(-(DoF+d)/2);
D=0;

c=(Block_1/Block_2)*Block_3;
```

emp_dis2.m

```
function X=emp_dis(Y)
% Compute the empirical marginal transformation, for the asset return
% samples, as described in Mashal(2002), "Beyond Correlation", pag.15
% As input we provide the i-th row of our d-columns sample
% i: #observation        d: #asset in the basket

X=zeros(length(Y),1);                % create a vector of d dimension
count=0;
for k=1:length(Y)
    for s=1:length(Y)
        if Y(s)<=Y(k)
            count=count+1;
        end
    end
    X(k)=count/length(Y);            % compute the empirical
                                     % distribution function
    if X(k)==1
        X(k)=.999999;                % otherwise, if 1, then we have
                                     % problem in using the t inverse
    end
    count=0;
end
```

kendall.m

```
function TAU=KENDALL(A,B);

% This function computes the Kendall's tau for a bivariate vector of
% observations

A=A(:);
B=B(:);
N=length(A);
N1=0;N2=0;S=0;
for J=1:N
    A1=A(J+1:N)-A(J);
    A2=B(J+1:N)-B(J);
    AA=A1.*A2;
    i=find(AA>=0);
    lni=length(i);
    k=find(~AA(i));                   % find the zero products.
    dis=length(AA)-lni;               % discordant pairs.
    con=lni-length(k);                % concordant pairs
    l=length(find(~(~A1(i(k)))));     % add up the extra zeros
    m=length(find(~(~A2(i(k)))));
    N1=N1+con+dis+l;
    N2=N2+con+dis+m;
    S=S+con-dis;
end
TAU=S/sqrt(N1*N2);
```

pseudo_sample.m

```
% Starting from a random sample M, this script creates a pseudo sample and
% compute the estimated correlation matrix using Kendall's Tau as described
% in Mashal(2001), "Beyond Correlation", p.41.

% T:    # of observation in the time series for each asset
% N:    # of assets

M = xlsread('C:\copula\DataSet_new.xls'); % equity price database
N=size(M,2);
T=size(M,1);
U=zeros(size(M));
corr_matrix=zeros(N);

% for each underlying asset we calculate the empirical distribution
for n=1:N
    U(:,n)=emp_dis(M(:,n));
end

% create the correlation matrix (via Kendall's tau), using the procedure
% described in page 42-42 of Mashal(2001) "Beyond Correlation"

for i=1:N
    for j=i:N
        if i==j
            corr_matrix(i,j)=1;
        else
        corr_matrix(i,j)=sin(pi*0.5*KENDALL(U(:,i),U(:,j)));
        corr_matrix(j,i)=corr_matrix(i,j);
        end
    end
```

```
end

CORR_EST=corr_matrix;
CORR_LIN=corrcoef(U);
```

表 2.2 中展示了对应的校准相关系数矩阵 $\boldsymbol{R}^{\mathrm{CML}}$.

表 2.2 校准的相关系数矩阵

1	0.448 18	0.902 08	0.839 75
0.448 18	1	0.676 15	0.685 52
0.902 08	0.676 15	1	0.841 78
0.839 75	0.685 52	0.841 78	1

2.6 Excel 中的 copula 应用

图 2.5 展示了使用二元 Frank copula 对一个投资组合的随机蒙特卡罗（Monte carlo）模拟法. 图 2.6 展示了一个对 Clayton copula 的模拟.

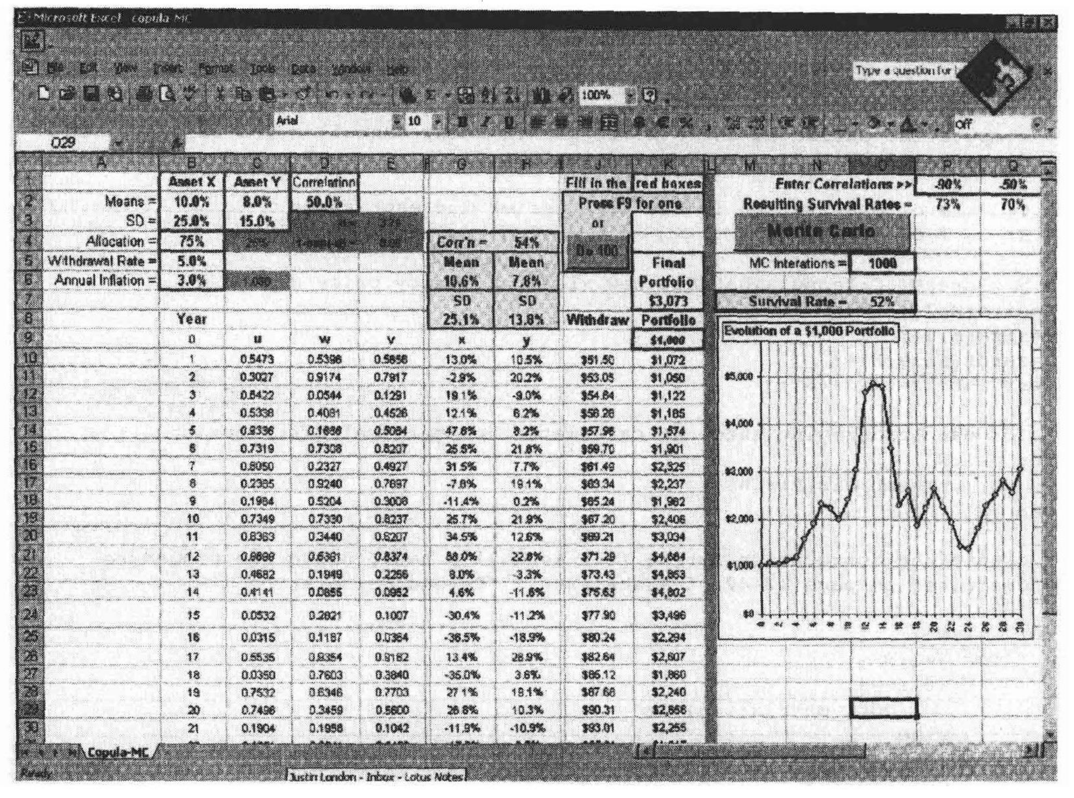


图 2.5 随机蒙特卡罗模拟法

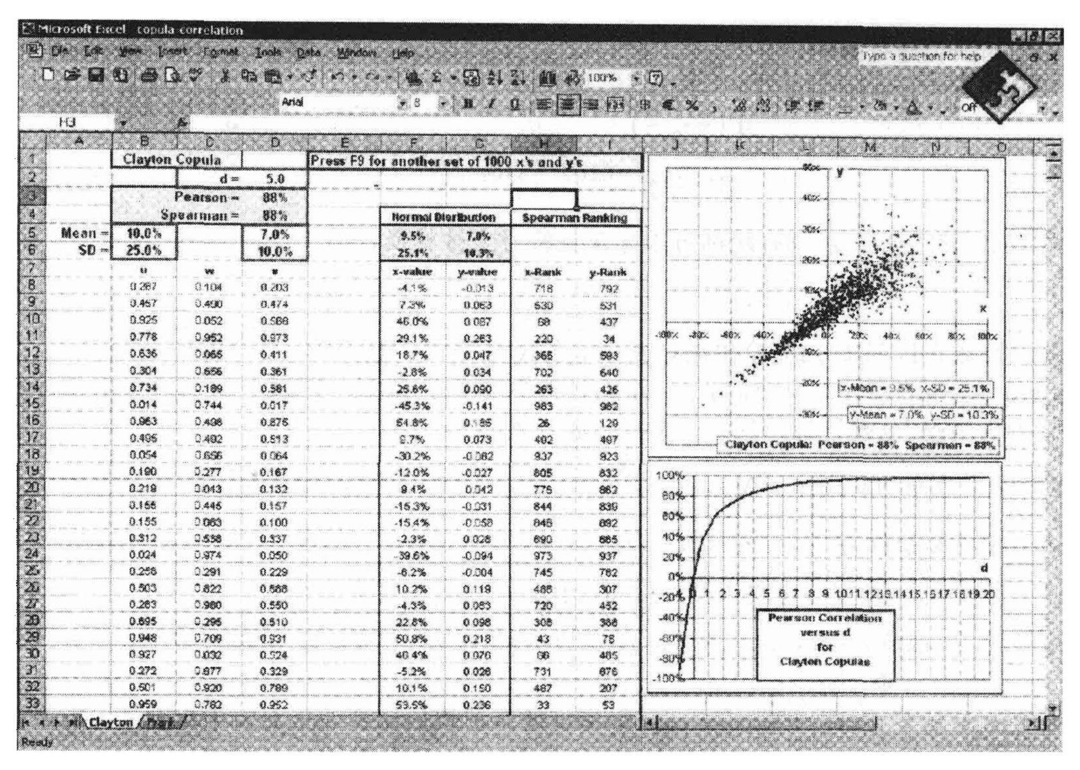


图 2.6　Clayton copula 模拟法

尾注

1. 一个 n 维盒子 $\boldsymbol{B}=[\boldsymbol{a}, \boldsymbol{b}]$ 的体积 $V_C(\boldsymbol{B})$ 定义为：

$$V_C(\boldsymbol{B}) = \sum_{\boldsymbol{d}} \operatorname{sgn}(\boldsymbol{d}) C(\boldsymbol{d})$$

$$= \sum_{i_1=1}^{2} \sum_{i_2=1}^{2} \cdots \sum_{i_n=1}^{2} (-1)^{i_1+i_2+\cdots+i_n} C(d_{1i_1}, d_{2i_2}, \cdots, d_{ni_n}) \geqslant 0$$

其中对于所有的 $j \in \{1, 2, \cdots, n\}$，$d_{j1}=a_j$，$d_{j2}=b_j$.

2. 在单变量情况下，一个随机变量 X 的密度函数 $f(x)$ 可以通过累积分布函数得到，关系如下：

$$f(x) = \frac{\partial F(x)}{\partial x}$$

3. 基于 Fang、Kotz 和 Ng(1987) 提出的概念，如果 $\boldsymbol{X}$ 是一个 n 维随机向量，并且对于 $\boldsymbol{\mu} \in \mathcal{R}^n$ 和一些 $n \times n$ 非负定对称矩阵 $\boldsymbol{\Sigma}$，$\boldsymbol{X}-\boldsymbol{\mu}$ 的特征函数 $\varphi_{X-\mu}(\boldsymbol{t})$ 是一个二次型函数 $\boldsymbol{t}'\boldsymbol{\Sigma}\boldsymbol{t}$，则 $\boldsymbol{X}$ 服从参数为 $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\varphi})$ 的椭圆分布，记作 $\boldsymbol{X} \sim \boldsymbol{E}^n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\varphi})$.

4. 根据 Johnson 和 Kotz(1972)，134 页，给定一个服从联合标准多元正态分布的随机向量 $\boldsymbol{X}=(X_1, \cdots, X_n)'$，其有相关矩阵 $\boldsymbol{R}$ 和一个独立于 $\boldsymbol{X}$ 的服从 χ_v^2 分布的随机变量 S. 我们将相关矩阵为 R、自由度为 v 的标准多元学生 t 分布的联合密度函数定义为随机向量 $\boldsymbol{Y}=\left(\frac{X_1}{S/\sqrt{v}}, \cdots,\right.$

$\frac{X_n}{S/\sqrt{v}})'$的联合分布函数：

$$f(y)=\frac{\Gamma\left(\frac{v+n}{2}\right)}{\Gamma\left(\frac{v}{2}\right)}\frac{1}{(\pi v)^{n/2}|R|^{1/2}}\left(1+\frac{y'R^{-1}y}{v}\right)^{-\frac{(v+n)}{2}}$$

5. 如果一个函数 $f(t)$ 对所有阶都可求导并且符号交替变换，即

$$(-1)^k\frac{\mathrm{d}^k}{\mathrm{d}t^k}f(t)\geqslant 0$$

则称该函数在区间 D 内完全单调，其中 t 在区域 D 内，且 $k=0, 1, 2, 3\cdots$

6. Mashal 和 Naldi(2002) 中提出了一个估计边际参数的程序，该程序基于单变量收益概率的数值优化例程. 也就是说，从一个单变量收益的抽样 $\{X_t\}_{t=1}^{T}$开始，构建一个服从自由度为 v 的学生 t 分布的新样本 $\{\hat{X}_t\}_{t=1}^{T}=\{\frac{X_t-m}{H}\}_{t=1}^{T}$，其中 m 为位置参数，H 是比例因子. 因此，给定一个参数空间 $\Theta=\{\theta: m\in\mathcal{R}, H>0, v>2\}$（其中 $\theta=(m, H, v)$，可以将极大似然估计量$\hat{\theta}=(\hat{m}, \hat{H}, \hat{v})$ 定义为$\hat{\theta}=\arg\max_{\theta\in\Theta}\prod_{t=1}^{T}f_v^{\text{student}}(\hat{X}_t)$.

7. 注意在等式（2.15）中，$\zeta=(t_v^{-1}(u_1), t_v^{-1}(u_2), \cdots, t_v^{-1}(u_N))'$. 同样，对于等式(2.13),$\zeta=(\Phi^{-1}(u_1), \cdots, \Phi^{-1}(u_N))'$.

8. 这一等式是从等式（2.15）中 t copula 密度的对数似然函数的最大值中得到的：

$$\frac{\partial\ell^{\text{Student}}(\theta)}{\partial R^{-1}}=\frac{T}{2}R-\frac{v+N}{2}\sum_{t=1}^{T}\frac{\frac{\xi_t'\xi_t}{v}}{\left(1+\frac{\xi_t'R^{-1}\xi_t}{v}\right)}$$

从中我们发现 R 的极大似然（ML）估计值必须满足：

$$R_{\text{ML}}=\frac{v+N}{Tv}\sum_{t=1}^{T}\frac{\xi_t'\xi_t}{\left(1+\frac{\xi_t'R_{\text{ML}}^{-1}\xi_t}{v}\right)}$$

9. 根据 Embrechts、Lindskog 和 McNeil(2001) 中所给出的定义，令（x, y）和($\tilde{x}$, $\tilde{y}$)为连续随机变量向量（X, Y）的两个观测值. 如果（$x-\tilde{x}$)$-$($y-\tilde{y}$)>0，那么（x, y）和（$\tilde{x}$, $\tilde{y}$）称为是一致的，否则非一致. Kendall's tau 被定义为一致的概率减去非一致的概率；即已知两对独立的随机变量（X, Y）和（$\widetilde{X}$, $\widetilde{Y}$）服从同一分布 $F(.,.)$，我们有

$$\tau=P[(X-\overline{X})(Y-\overline{Y})>0]-P[(X-\overline{X})(Y-\overline{Y})<0]$$

如 Meneguzzo 和 Vecchiato(2002) 中建议的，已知两个数列 X_t 和 Y_t，$t=1, \cdots., T$，Kendall's tau 的一致估计值的计算如下：

$$\tau=\frac{2}{T(T-1)}\sum_{i<j}\text{sgn}[(X_i-X_j)(Y_i-Y_j)]$$

其中

$$\text{sgn}(x)=\begin{cases}1, \text{若 } x\geqslant 0\\ -1, \text{若 } x<0\end{cases}$$

第3章 住房抵押贷款证券

住房抵押贷款证券（MBS）和抵押转付证券（Pass-Throughs，PT）是对住房抵押贷款组合的索偿权.是这样创建的：一个联邦机构、住房抵押贷款融资机构、银行或者投资公司购买某种类型的住房抵押贷款——比如，由联邦住房管理机构（Federal Home Administration，FHA）或者退伍军人管理机构（Veterans' Administration，VA）担保的住房抵押贷款，之后再将对投资组合现金流的索债权以MBS的形式出售，为购买住房抵押贷款融资.MBS有两种形式：机构形式（agency）和常规（conventional）形式（私募）[1].

机构形式的MBS（比如GNMA转付证券）是拥有对由FHM、VA或FmHA(Farmers Martgage Home Adminishtration)担保违约风险的住房抵押贷款组合的索债权的证券.住房抵押贷款融资机构、银行或者投资公司向GNMA(Ginnie Mae)提供一批特定类型（30年固定利率或15年浮动利率）的FHA、VA或FmHA担保.如果这批担保是适当的（in order），GNMA会发行一个单独的保证，从而允许这批担保的MBS以GNMA PT形式发行.其他机构形式的MBS（包括联邦住房贷款抵押公司（Federal Home Loan Mortgage Corporation，FHLMC）的MBS）是对常规住房抵押贷款组合的索偿权.FHLMC通过从抵押贷款发放者手中购买抵押贷款，生成一个被称为*参与凭证*（participation certificate）的MBS，然后通过经销商网络来发行机构形式的MBS.FHLMC有一个互换项目，通过这个项目，FHLMC将MBS互换为存款和贷款或者特定类型的商业银行抵押贷款组合.其他政府机构（如FNMA（Fannie Mae））发行多种类型的MBS：参与凭证、互换和PT.通过这些凭证，业主的还款从抵押贷款发放银行通过发行机构到凭证持有者手中.

常规形式的MBS（或称专用标签形式的MBS）是由商业银行（通过其控股公司）、S&Ls、抵押贷款融资机构和投资公司发行的.常规形式的MBS包括那些由Prudential Home、Chase Mortgage、CitiCorp Housing、Ryland/Saxon、GE Capital和Countrywide发行的MBS.常规形式的PT必须在证券交易委员会（SEC）注册.这些PT通常是由私募发行者以信用证（LOC）形式的外部保险以及开发初级和高级PT的内部保险来担保的.

MBS存在一级市场和二级市场.一级市场上的投资者自己直接购买或通过经纪商购买机构和私募投资公司发行的MBS.很多投资者都是机构投资者.因此，MBS的创造为机构投资者提供了一个房地产投资的工具.在一级市场中，MBS的发行量主要在25 000到250 000美元之间（一部分可高达100万美元），其中一部分可赎回.在二级市场中，MBS在场外（Over-The-Counter，OTC）进行交易.场外经销商都是住房抵押贷款证券交易商协会（Mortgage-Backed Securities Dealer Association，MSDA）的会员.

由于MBS对提前偿还和利率的敏感性，它是最难进行建模和估价的一种证券，这种敏感性会影响到现金流流向投资者的时间、频率和大小.MBS的现金流来自抵押贷款组合中的每月现金流.现金流中包括本金所产生的利息、计划内本金和提前偿还的本金.介于现金流对相关MBS特性的影响力，现金流分析是对任何MBS估价的基础，其中包括*加权平均到期日*（Wetghted Average Maturity，WAM）、*加权平均票面*（Weighted Average Coupon，WAC）利率、

转付（PT）率和提前偿还率（Prepaymet rate）或速率（Speed）. WAM是全部预计支付给投资者的MBS现金流的有效久期，或称时间加权. WAC是一个用来决定计划内本金的抵押贷款组合利率. PT率是本金利率并且低于WAC，两者间的差值归MBS的发行者所有. 预偿还率或速率是由抵押贷款的业主决定的提前偿还率.

本章将详细讨论MBS的定价和建模. 3.1节将讨论MBS定价的提前偿还和PSA模型. 3.2节将给出数值例子来解说提前偿还模型的工作如何在Excel中实现. 3.3节将讨论MBS的定价和报价，还有基于不同的利率假设条件下的投资者回报. 3.4节将讨论提前偿还的风险和MBS的平均寿命. 3.5节会详细复习运用蒙特卡罗模拟进行估价分析和现金流分析及其数值应用. 3.6节会给出使用Matlab中固定收益工具包的数值例子. 3.7节将讨论MBS衍生品，其中包括住房抵押贷款债券（CMO）和顺序支付额度结构. 我们会给出使用Excel计算的例子. 3.8节中讨论CMO在C++中的实现. 3.9节将讨论计划摊还证券（PAC）及其结构. 在3.10节中，我们会回顾带状MBS，其中包括纯利息证券（IO）和纯本金证券（PO）. 在3.11节中，我们将讨论MBS的利率风险. 最后，在3.12节里，我们会讨论MBS的对冲和如何使用MBS来进行资产负债表的资产负债管理.

3.1 提前偿还模型

MBS估价模型通常假定一个提前偿还率或速率. 尽管大多数模型都源于一个基准模型或基准利率，但在分析MBS时，投资者和发行者会使用不同的提前偿还模型. 这一基准模型是由公共证券协会（PSA）提出的. PSA使用条件提前偿还率（Conditional Prepayment Rate，CPR）来衡量速率. CPR是每月提前偿还金额与抵押贷款余额的比率，以年为单位报价. 这一月利率称为单月死亡率（SMM）：

$$\mathrm{SMM} = 1 - (1 - \mathrm{CPR})^{1/12} \tag{3.1}$$

预计月度提前偿还金额为：

$$\text{月度提前偿还金额} = \mathrm{SMM} \times [\text{月度初始金额} - \text{本月预计偿还本金}]$$

举例说明，若CPR=6%，月度初始金额=\$100M，而本月预计支付本金=\$3M，则预计月度提前偿还金额为\$0.499M：

$$\mathrm{SMM} = 1 - [1 - 0.06]^{1/12} = 0.005\,143$$

$$\text{月度提前偿还本金} = 0.005\,143[\$100\mathrm{M} - \$3\mathrm{M}] = \$0.499\mathrm{M}$$

在PSA模型中，CPR依赖于抵押贷款的到期时间. PSA的标准模型假设对于30年（360个月）的抵押贷款，CPR在第1个月为0.2%，之后在30个月内增长到6%，然后保持6%直到抵押贷款到期. 这一模型称为100%PSA模型. 图3.1描述了提前偿还率关于月份的函数.

t月的CPR估计值为：

$$\mathrm{CPR} = \begin{cases} 0.06\left(\dfrac{t}{30}\right), & \text{若 } t \leqslant 30 \\ 0.06, & \text{若 } t > 30 \end{cases} \tag{3.2}$$

例如，第 5 个月的 CPR 为：

$$\mathrm{CPR} = 0.06\left(\frac{5}{30}\right) = 0.01$$

$$\mathrm{SMM} = 1 - [1 - 0.01]^{1/12} = 0.000\,837$$

通过将标准模型（100%PSA）表示为或高或低的百分比，譬如 150%或 50%，PSA 模型可以用不同的速率来定义. 在低速率时段，PSA 模型可以达到 150%，在高速率时段，它可以达到 50%. 对于 100%PSA 模型，30 年期抵押贷款的平均持有期为 17 年；对于 225%PSA 模型则是 8 年. 图 3.2 描述了不同提前偿还率关于月份的函数.

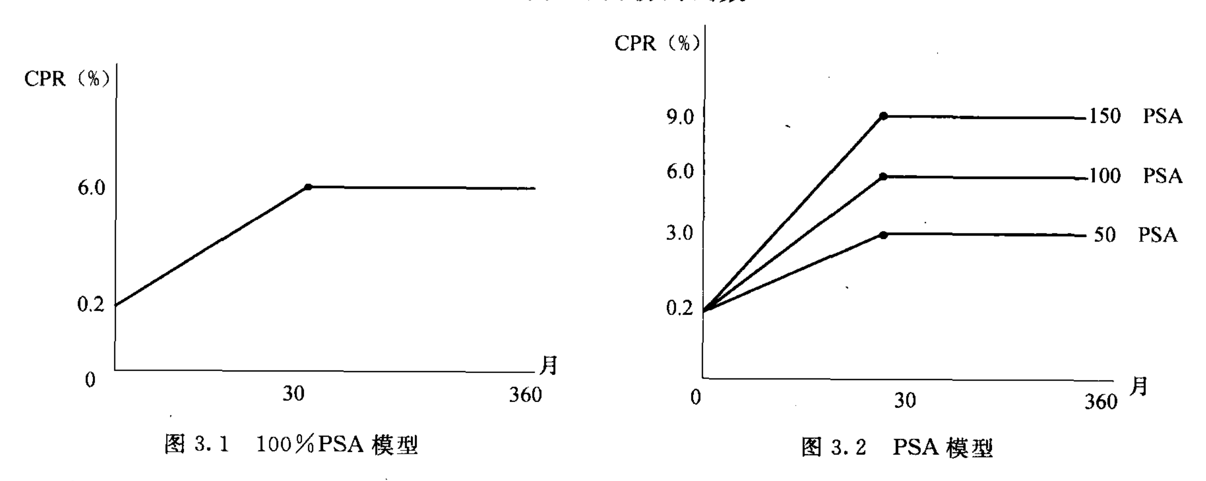


图 3.1 100%PSA 模型

图 3.2 PSA 模型

假设我们想要计算在 165PSA 速率下第 5 个月的 CPR 和 SMM，可以根据式（3.1）和式（3.2）计算：

$$\mathrm{CPR} = 0.06\left(\frac{5}{30}\right) = 0.01$$

$$165\mathrm{CPR} = 1.65(0.01) = 0.016\,5$$

$$\mathrm{SMM} = 1 - [1 - 0.016\,5]^{1/12} = 0.000\,138\,6$$

3.2 提前偿还模型的数值例子

令 p=月度计划偿还本金，F_0=MBS 的标的抵押贷款的面值，M=WAM=到期前的加权平均月数，I=偿还利率，SP=计划偿还本金，PP=提前偿还本金，R_A=年度利率（WAC），$B_i(i=1, \cdots, 360)$ 表示第 i 月的抵押贷款余额，然后 $CF_i(i=1, \cdots, 360)$ 表示第 i 月的现金流. 注意，第 1 个月的抵押贷款余额是 MBS 的初始面额，$B_1=F_0$. 以下公式给出了 p：

$$p = \frac{F_0}{\left(\frac{1-1/(1+(R_A/12))^M}{R_A/12}\right)} \tag{3.3}$$

考虑一个面额为 10 000 万美元的住房抵押贷款组合，WAC=9%，WAM=360 个月，提前偿还速率=100%PSA. 需要计算第 1 个月的多个现金流. 第 1 个月的计划偿还本金为：

$$p = \frac{\$100M}{\left(\frac{1-1/(1+(0.09/12))^{360}}{0.09/12}\right)} = \$804\,600$$

利息支付为：

$$I = \left(\frac{0.09}{12}\right)\$100M = \$750\,000$$

计划偿还本金为：

$$SP = \$804\,600 - \$750\,000 = \$54\,600$$

第 1 个月的预计提前偿还本金为：

$$\mathrm{CPR} = \left(\frac{1}{30}\right)0.06 = 0.002$$

$$\mathrm{SMM} = 1-[1-0.002]^{1/12} = 0.000\,166\,8$$

$$PP = 0.000\,166\,8[\$100M - \$54\,620] = \$16\,667$$

第 1 年的现金流为：

$$CF_1 = p + PP + I$$

$$= \$804\,600 + \$750\,000 + \$16\,667 = \$821\,295$$

第 2 个月的期初余额为：

$$B_2 = B_1 - SP - PP$$

$$= \$100M - \$54\,600 - \$16\,667M = \$99.929M$$

第 2 个月的现金流计算如下. 第 2 个月的偿还金额为：

$$p = \frac{\$99.928\,7M}{\left[\frac{1-1/(1+(0.09/12))^{359}}{0.09/12}\right]} = \$804\,488$$

利息支付为：

$$I = \left(\frac{0.09}{12}\right)\$99.928\,7M = \$749\,465$$

计划偿还本金为：

$$SP = \$804\,488 - \$749\,465 = \$55\,023$$

预计提前偿还本金为：

$$\mathrm{CPR} = \left(\frac{2}{30}\right)0.06 = 0.004$$

$$\mathrm{SMM} = 1-[1-0.004]^{1/12} = 0.000\,333\,9$$

$$PP = 0.000\,333\,0[\$99.928\,7M - \$55\,023] = \$33\,352$$

因此，第 2 个月的现金流为：

$$CF_2 = \$749\,400 + \$55\,023 + \$33\,352 = \$837\,840$$

其余月份的现金流以类似方式计算.

考虑一个面值为 \$100M 的住房抵押贷款组合，WAC=8.125%，WAM=357 个月，PT 率=7.5%，提前偿还速率=165%PSA. 注意到，因为 WAM 不是 360 个月，而是 357 个月，资产寿命是按季度计算的，所以还款实际上是从第 4 个月开始的，而非第 1 个月.

此外，我们用 PT 率来计算利息支付. 但是，我们用 WAC 率来计算计划偿还本金和抵押贷款偿还金额.

在这个例子中，计划月度抵押贷款偿还金额为：

$$p = \frac{\$100M}{\left[\frac{1-1/(1+(0.081\,25/12))^{357}}{0.081\,25/12}\right]} = \$743\,970$$

利息支付（使用 PT 率计算）为：

$$I = \left(\frac{0.075}{12}\right)\$100M = \$625\,000$$

计划偿还本金（使用 WAC 率）为：

$$SP = \$743\,970 - (0.081\,25/12)(\$100M) = \$66\,880$$

使用 165%PSA，预计提前偿还本金为：

$$CPR = 1.65\left(\frac{4}{30}\right)0.06 = 0.013\,2$$

$$SMM = 1 - [1 - 0.013\,2]^{1/12} = 0.001\,106\,7$$

$$PP = 0.001\,106\,7[\$100M - \$66\,880] = \$110\,600$$

第 1 个月的现金流（从第 4 个月开始）为：

$$CF_1 = \$625\,000 + \$66\,880 + \$110\,600 = \$802\,480$$

第 2 个月的期初抵押贷款余额为：

$$\$100M - \$66\,880 - \$110\,600 = \$99.822M$$

第 2 个月的计划月度抵押贷款偿还金额为：

$$p = \frac{\$99.822M}{\left[\frac{1-1/(1+(0.081\,25/12))^{356}}{0.081\,25/12}\right]} = \$743\,140$$

第 2 个月的利息支付为：

$$I = \left(\frac{0.075}{12}\right)\$99.822M = \$623\,890$$

第 2 个月的计划偿还本金为：

$$SP = \$743\,140 - (0.081\,25/12)(\$99.822M) = \$67\,260$$

第 2 个月的预计提前偿还本金为：

$$CPR = 1.65\left(\frac{5}{30}\right)0.06 = 0.016\,5$$

$$SMM = 1 - [1 - 0.016\,5]^{1/12} = 0.001\,39$$

$$PP = 0.001\,89[\$99.822M - \$67\,260] = \$138\,210$$

因此，第 2 个月的现金流为：

$$CF_2 = \$623\,890 + \$67\,260 + \$138\,210 = \$829\,360$$

表 3.1 显示了前几个月的现金流.

表 3.1

月	月初余额	计划偿还金额	利息	计划偿还本金	CPR	SMM	提前偿还本金	CF	月末余额
1	100 000 000.00	743 967.06	625 000.00	66 883.73	0.013 2	0.001 106 7	110 597.15	802 480.87	99 822 519.13
2	99 822 519.13	742 153.62	623 890.74	66 271.98	0.016 5	0.001 385 5	138 213.22	828 375.94	99 618 033.93
3	99 618 033.93	740 145.25	622 612.71	65 648.15	0.019 8	0.001 665 2	165 771.24	854 032.10	99 386 614.54
4	99 386 614.54	737 942.81	621 166.34	65 012.61	0.023 1	0.001 945 7	193 248.74	879 427.69	99 128 353.20
5	99 128 353.20	735 547.31	619 552.21	64 365.76	0.026 4	0.002 227 1	220 623.21	904 541.17	98 843 364.23
6	98 843 364.23	732 959.93	617 771.03	63 707.98	0.029 7	0.002 509 3	247 872.18	929 351.19	98 531 784.07
7	98 531 784.07	730 181.99	615 823.65	63 039.70	0.033 0	0.002 792 5	274 973.22	953 836.56	98 193 771.16
8	98 193 771.16	727 214.96	613 711.07	62 361.30	0.036 3	0.003 076 5	301 903.97	977 976.35	97 829 505.88

资料来源：Johnson S(2004).

Excel 表格 MBS1. xls 显示了对每个月的完整计算. 可以通过更改参数（针对不同的假设）来得到不同的现金流.

3.3 住房抵押贷款证券的定价和报价

MBS 的价格是以标的住房抵押贷款余额的比率形式报出的. t 时刻的住房抵押贷款余额 F_t 是以初始金额的比率形式报出的. 这被称为组合因子（pool factor）pf_t：

$$pf_t = \frac{F_t}{F_0} \tag{3.4}$$

举例来说，由初始价值为 \$100M 的抵押贷款组合来担保的 MBS，pf 现值为 0.92，报价为 95－16（注：16 表示 16/32），市场价值为 \$87.86M，计算如下：

$$\begin{aligned} F_t &= (pf_t)F_0 \\ &= (0.92)(\$100\text{M}) = \$92\text{M} \end{aligned}$$

因此，

$$\text{市场价值} = (0.9550)(\$92\text{M}) = \$87.86\text{M}$$

市场价值为净价，不将累计利息计算在内，记作 AI. 对于 MBS，累计利息是基于从结算日（交易发生的两天后）到下个月的第 1 天这段时期的. 举例来说，若这段时间是 20 天，而一个月是 30 天，WAC=9%，则 AI 是 \$0.46M：

$$AI = \left(\frac{20}{30}\right)\left(\frac{0.09}{12}\right)\$92\text{M} = \$460\,000$$

全部市场价值为 \$88.32M：

$$\text{全部市场价值} = \$87.86\text{M} + \$460\,000 = \$88.32\text{M}$$

每股市价等于全部市场价值除以股份数. 若股份数为 400，则基于 95－16 报价的 MBS 的价格是 \$220 080：

$$\text{MBS 价格} = \frac{\$88.32\text{M}}{400} = \$220\ 800$$

MBS 的价值等同于证券现金流的现值（PV）. 因此，这个价值是 MBS 的期望现金流和利率的函数. 另外，对于 MBS，现金流也与利率 R 相关：利率变化会导致本金提前偿还金额的变化，并且增加或减少早期现金流：

$$V_{\text{MBS}} = f(CF, R)$$

其中，$CF=f(R)$.

由于现金流是关于利率的函数，相对于类似的公司债券，MBS 价值对利率变化更敏感. 这一敏感性就是延期风险（extension risk）. 注意下面的关系：

$$\text{若 } R\downarrow \Rightarrow \text{贴现率下降} \Rightarrow V_M\uparrow \text{（与任何其他的债券一样）}$$

而且，

$$\text{若 } R\downarrow \Rightarrow \text{提前偿还金额增加：} \Rightarrow V_M\uparrow$$
$$\Rightarrow \text{早期现金流} \uparrow$$

另一方面，

$$\text{若 } R\uparrow \Rightarrow \text{贴现率上升} \Rightarrow V_M\downarrow$$

而且，

$$\text{若 } R\uparrow \Rightarrow \text{提前偿还金额下降：} \Rightarrow V_M\downarrow$$
$$\Rightarrow \text{早期现金流} \downarrow$$

所以利率的上升会使 MBS 的市场价值下跌，从而导致延期风险.

除了再融资率之外，还存在很多影响提前偿还金额的外在因素和内在因素。其中之一就是住房换手率（housing turnover）——借款人通过出售房子提前还贷的比率. 另外一个就是调整时期，自愿提前还款金额（住房换手率、再融资兑现和信用升级，但不含再融资或违约率）升至长期水平所需要的月数. 其他一些因素中包括贷款利率调降（credit curing）——由于信用等级上升或房产价值增加使得借款人获得更优惠的利率或更高的贷款金额而导致的长期贷款利率下降. 当抵押贷款逐渐耗尽，贷款利率调降也会降低[2]. 由 PSA 标准违约假设（SDA）的一个比率来表示的违约和消耗的最大利率相关 CPR 都会影响提前偿还金额——抵押贷款如果没有经历过相关优先利率的再融资，那么其 CPR 会较低. 这一比例越低，抵押贷款消耗得越快[3].

很多华尔街的公司使用专用精简形式的提前偿还模型（proprietary reduced-form prepayment），这个模型使用过去的提前偿还率和内生变量来解释提前偿还. 这些模型经过校准来拟合观测数据，不受理论考量的制约[4].

3.4 提前偿还风险和 MBS 的平均寿命

平均寿命（average life）是指 MBS 抵押贷款期限的加权平均，其权重为定期现金流数额与本金总额的比例. 举例来说，30 年、$100M、9%、100PSA 的抵押贷款组合（3.2 节中的第 1 个例子）的初始平均寿命为 12.077 年，计算如下：

$$\text{平均寿命} = \frac{1}{12}\,\frac{(1(\$71\ 295) + 2(\$88\ 376) + \cdots + 360(\$135\ 281))}{\$100\ 000\ 000} = 12.077$$

一般来说，MBS 平均寿命的计算公式如下：

$$\text{平均寿命} = \frac{1}{12}\frac{\sum_{i=1}^{360} i * CF_i}{F_0} \tag{3.5}$$

提前偿还风险可由 MBS 抵押贷款的平均寿命对提前偿还速率的变化（PSA 的变化）或等同于利率的变化（因为利率变化是速率变化的重要因素）的敏感性来衡量：

$$\text{提前偿还风险} = \frac{\Delta\ \text{平均寿命}}{\Delta \text{PSA}} \cong \frac{\Delta\ \text{平均寿命}}{\Delta R} \tag{3.6}$$

如果提前偿还风险$=\frac{\Delta\ \text{平均寿命}}{\Delta \text{PSA}}=0$，那么 MBS 或其抵押品不会出现提前偿还.

MBS 衍生品的发展，如计划摊还证券（Planned Amortization Classes，PAC）是 20 世纪 80 年代最具影响力的金融创新之一，它拥有不同的提前偿还风险性质，其中还包括一些无提前偿还风险的衍生品.

提前偿还率的假设基于再融资利率的变化概率. 举例来说，如果美联储（Federal Open Markets Committee（the Fed））降低利息（比如基于媒体报道）的可能性较大，我们可以期待再融资率会随之下跌，从而业主可以用更低的利率再融资. 这样会导致提前偿还的速率和投资者手中的现金流增加. PSA 率从而会随之上升. 相反，如果美联储为了应对通货膨胀而提高利率，预期再融资利率会上升，提前还款风险会降低，MBS 的平均寿命会延长. PSA 率从而会下调.

蒙特卡罗模拟是对再融资率方案进行建模的最好途径. 我们首先构建一棵利率树——如二叉树[5]——在每个交叉点上都有即期利率和再融资利率. 我们可以按照树中可能的利率路线来运行很多个模拟路线. 对于任何一个模拟路线，我们都要基于这条路线上的每一个时间点的再融资利率来估计其现金流.（路线上的每一个时间点都与短期利率树上的时间点相符合.）通过以下的步骤，蒙特卡罗模拟可以被用来决定 MBS 的理论值或收益率：

1. 模拟利率. 用二叉利率树来得到即期利率和再融资利率的不同路线.
2. 基于即期利率来确定一个提前偿还模型.
3. 形成一条抵押贷款组合、MBS 或份额（tranche）的现金流路线.
4. 确定每条路线的现值、路线的分布平均值（理论值）和标准差；或者给定市场价值，确定每条路线的收益率、分布、平均值和标准差.

第 1 步

在第 1 步中，为了模拟利率，我们从一棵二叉利率树中生成利率路线[6]. 例如，假定一个一年期即期利率的三期二叉树 R_t^S，再融资利率为 R_t^{ref}，其中 $R_0^S=6\%$，$R_0^{\text{ref}}=8\%$，$u=1.1$，$d=0.9091=1/1.1$. 三期后，二叉树会有 4 个可能的利率，8 条可能的路线，如图 3.3 所示. 表 3.2 描述了从二叉树中模拟得到的 8 条短期利率路线.

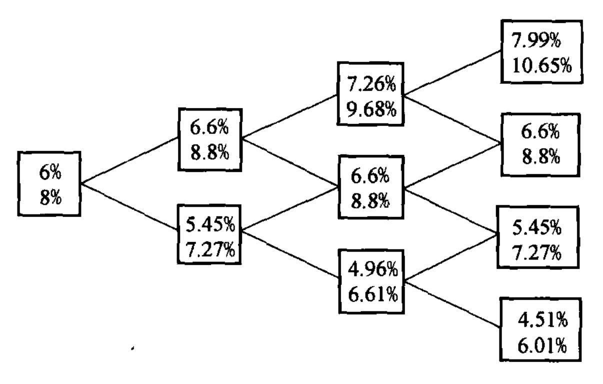


图 3.3 即期利率与再融资利率的二叉树

表 3.2

	第 1 年（%）	第 2 年（%）	第 3 年（%）	第 4 年（%）
路线 1	8.000 0	7.272 8	6.611 7	6.010 7
路线 2	8.000 0	7.272 8	6.611 7	7.272 8
路线 3	8.000 0	7.272 8	8.000 0	7.272 8
路线 4	8.000 0	8.800 0	8.000 0	7.272 8
路线 5	8.000 0	7.272 8	8.000 0	8.000 0
路线 6	8.000 0	8.800 0	8.000 0	8.000 0
路线 7	8.000 0	8.800 0	9.680 0	8.800 0
路线 8	8.000 0	8.800 0	9.680 0	10.648 0

假设我们有一个面值为＄100 万美元的抵押贷款组合，WAM＝10 年，WAC＝8%，PT 率＝8%，以年为单位派发现金流，抵押贷款不存在违约风险，而且在第 4 年末有一个大额尾付金额 (balloon payment)，等于第 4 年初的贷款余额（比如第 4 年的计划本金）. 我们可以用如下方法计算计划每月的抵押贷款偿还金额：

$$p = \frac{\$1\,000\,000}{\dfrac{1-(1/1.08)^{10}}{0.08}} = \$149\,029$$

如果初始假定没有提前还款风险，则我们可以得到表 3.3 中所示的现金流. 第 4 年末的大额尾付金额为：

表 3.3

年 份	余 额	价 格	利 息	本 金	现金流量（CF）
1	＄1 000 000	＄149 029	＄80 000	＄69 029	＄149 029
2	＄930 970	＄149 029	＄74 478	＄74 551	＄149 029
3	＄856 419	＄149 029	＄68 513	＄80 516	＄149 029
4	＄775 903	＄149 029	＄62 072	＄86 957	＄837 975

$$\begin{aligned}\text{大额尾付金额} &= \text{贷款余额(第 4 年)} - \text{计划本金(第 4 年)} \\ &= \$775\,149 - \$86\,957 = \$688\,946\end{aligned}$$

第 4 年的现金流为：

$$\begin{aligned}CF_4 &= \text{大额尾付金额} + p \\ &= \$688\,946 + \$149\,029 = \$837\,973\end{aligned}$$

或者相当于：

$$\begin{aligned}CF_4 &= \text{贷款余额(第 4 年)} + \text{利息} \\ &= \$775\,903 + \$62\,513 = \$837\,973\end{aligned}$$

第 2 步

蒙特卡罗模拟的第 2 步是建立一个提前还款模型. 列出提前还款安排，如表 3.4 所示. CPR 由差价 $X = WAC - R^{ref}$ 来决定.

表 3.4

范 围					CPR
		X	⩽	0	5%
0	<	X	⩽	0.5%	10%
0.5%	<	X	⩽	1.00%	20%
1.00%	<	X	⩽	1.25%	30%
1.25%	<	X	⩽	2.0%	40%
2.0%	<	X	⩽	2.5%	50%
2.5%	<	X	⩽	3.0%	60%
		X	>	3.0%	70%

第 3 步

估价过程的第 3 步是基于再融资利率的模拟路线、差价 X 和 CPR（见表 3.5）来估计每条路线的现金流.

表　3.5

路线 1 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 *CF*
1	0.0727 28	1 000 000	0.08	80 000	69 029.49	0.2	186 194.1	335 223.6
2	0.066 117	744 776.4	0.08	59 582.11	59 641.48	0.4	274 054	393 277.6
3	0.060 107	411 081	0.08	32 886.48	38 647.68	0.4	148 973.3	220 507.5
4		223 460	0.08	17 876.8				241 336.8
路线 2 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 *CF*
1	0.072 728	1 000 000	0.08	80 000	69 029.49	0.2	186 194.1	335 223.6
2	0.066 117	744 776.4	0.08	59 582.11	59 641.48	0.4	274 054	393 277.6
3	0.072 728	411 081	0.08	32 886.48	38 647.68	0.2	74 486.66	146 020.8
4		297 946.6	0.08	23 835.73				321 782.4
路线 3 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 *CF*
1	0.072 728	1 000 000	0.08	80 000	69 029.49	0.2	186 194.1	335 223.6
2	0.08	744 776.4	0.08	59 582.11	59 641.48	0.05	34 256.75	153 480.3
3	0.072 728	650 878.2	0.08	52 070.25	61 192.16	0.2	117 937.2	231 199.6
4		471 748.8	0.08	37 739.91				509 488.7
路线 4 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 *CF*
1	0.088	1 000 000	0.08	80 000	69 029.49	0.05	46 548.53	195 578
2	0.08	884 422	0.08	70 753.76	70 824.26	0.05	40 679.89	182 257.9
3	0.072 728	772 917.8	0.08	61 833.43	72 665.69	0.2	140 050.4	274 549.5
4		560 201.7	0.08	44 816.14				605 017.9
路线 5 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 *CF*
1	0.072 728	1 000 000	0.08	80 000	69 029.49	0.2	186 194.1	335 223.6
2	0.08	744 776.4	0.08	59 582.11	59 641.48	0.05	34 256.75	153 480.3
3	0.088	650 878.2	0.08	52 070.25	61 192.16	0.05	29 484.3	142 746.7
4		560 201.7	0.08	44 816.14				605 017.9
路线 6 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 *CF*
1	0.088	1 000 000	0.08	80 000	69 029.49	0.05	46 548.53	195 578
2	0.08	884 422	0.08	70 753.76	70 824.26	0.05	40 679.89	182 257.9
3	0.088	772 917.8	0.08	61 833.43	72 665.69	0.05	35 012.61	169 511.7
4		665 239.5	0.08	53 219.16				718 458.7

（续）

路线 7 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 CF
1	0.088	1 000 000	0.08	80 000	69 029.49	0.05	46 548.53	195 578
2	0.096	884 422	0.08	70 753.76	70 824.26	0.05	40 679.89	182 257.9
3	0.088	772 917.8	0.08	61 833.43	72 665.69	0.05	35 012.61	169 511.7
4		665 239.5	0.08	53 219.16				718 458.7
路线 8 年	1 再融资利率	2 余额	WAC	3 利息	4 计划本金	5 CPR	6 提前偿还本金	7 CF
1	0.088	1 000 000	0.08	80 000	69 029.49	0.05	46 548.53	195 578
2	0.096	884 422	0.08	70 753.76	70 824.26	0.05	40 679.89	182 257.9
3	0.106 48	772 917.8	0.08	61 833.43	72 665.69	0.05	35 012.61	169 511.7
4		665 239.5	0.08	53 219.16				718 458.7

资料来源：Johnson S(2004)．

下面给出第 1 条路线的现金流的计算过程．在第 1 年，抵押贷款计划偿还金额为：

$$p=\frac{\$1\,000\,000}{\dfrac{1-(1/1.08)^{10}}{0.08}}=\$149\,029$$

利息支付为：

$$I=0.08(\$1\,000\,000)=\$80\,000$$

计划本金为：

$$SP=\$149\,029-\$80\,000=\$69\,029$$

提前偿还本金金额为：

$$PP=0.20(\$1\,000\,000-\$69\,029)=\$186\,194$$

第 1 年的现金流为：

$$CF_1=\$80\,000+\$69\,029+\$186\,194=\$335\,223$$

对于第 2 年，根据路线 1，可得贷款余额为：

$$B_2=\$1\,000\,000-\$69\,029-\$186\,194=\$744\,776$$

每月抵押贷款计划还款额为：

$$p=\frac{\$744\,446}{\dfrac{1-(1/1.08)^{9}}{0.08}}=\$119\,223$$

利息支付为：

$$I=0.08(\$744\,776)=\$59\,582$$

计划偿还本金金额为：

$$SP=\$119\,223-\$59\,582=\$59\,641$$

第 2 年提前偿还本金金额为：

$$PP=0.4(\$755\,776-\$59\,641)=\$274\,054$$

现金流为：

$$CF_2=\$59\,582+\$59\,641+\$274\,052=\$393\,277$$

对于路线 1 的第 3 年，贷款余额为：

$$B_3 = \$744\,776 - \$59\,641 - \$274\,054 = \$411\,081$$

每月抵押贷款计划偿还金额为：

$$p = \frac{\$411\,081}{\frac{1-(1/1.08)^8}{0.08}} = \$71\,543$$

利息支付为：

$$I = 0.08(\$411\,081) = \$32\,886$$

计划偿还本金金额为：

$$SP = \$71\,543 - \$32\,886 = \$38\,648$$

第 3 年提前偿还本金金额为：

$$PP = 0.4(\$411\,081 - \$38\,648) = \$148\,973$$

现金流为：

$$CF_3 = \$32\,886 + \$38\,648 + \$148\,973 = \$220\,507$$

最后，对于路线 1 的第 4 年，贷款余额为：

$$B_4 = \$411\,081 - \$38\,648 - \$148\,943 = \$223\,460$$

利息支付为：

$$I = 0.08(\$223\,460) = \$17\,877$$

现金流为：

$$\begin{aligned} CF_4 &= B_4 + I \\ &= \$223\,460 + \$17\,877 = \$241\,337 \end{aligned}$$

所有其他路线上的现金流可类似计算.

第 4 步

估值过程的第 4 步是对每条路线上的现金流进行估值. 每条路线上现金流的现值都由特定的近似贴现率来决定. 因为抵押贷款不存在违约风险，所以投资者只承担提前还款风险. 此类风险的风险溢价称为*期权调整价差*（Option Adjusted Spread，OAS）. OAS 是在考虑了所有嵌入期权之后得到的对由 MBS 提供的国债利率的差价衡量[7]. 我们可以把 OAS 看做非模型风险（无法用模型表示的风险，比如对提前还款的估计错误）的市场价格. OAS 是这样一种差价：它累计所有利率路线上的即期利率，再使路线的平均现值等同于观测的市价（加上累计利息）. 因此，观测的证券市价等于它的理论值. 数学上等于下面公式中的 K 值：

$$\begin{aligned} P^{\text{Market}} &= \frac{1}{N}[PV(\text{路线 }1) + PV(\text{路线 }2) + \cdots + PV(\text{路线 }N)] \\ &= \frac{1}{N}\left[\sum_{i=1}^{T}\frac{CF_i^{\text{路线}1}}{(1+Z_i^1+K)^i} + \sum_{i=1}^{T}\frac{CF_i^{\text{路线}2}}{(1+Z_i^2+K)^i} + \cdots + \sum_{i=1}^{T}\frac{CF_i^{\text{路线}N}}{(1+Z_i^N+K)^i}\right] \end{aligned} \tag{3.7}$$

其中 Z_i^j 是在路线 $j=1, \cdots, N$ 上的时间 i(如 $i=1, \cdots, T$ 月）的 0 利率. 通常，$T=360$，$N=1\,024$.

现金流收益是一个对任何 MBS 的标准衡量，是*静态差价*（static spread）. 这是一个“理论性国债即期利率曲线而非国债收益曲线上单个点的静态债券系列（非波动利率）”的收益差

价[8]. 差价的大小依赖于收益曲线的陡峭程度：曲线越陡峭，债券收益和国债收益之间的差越大[9]. 计算静态差价有两种方法. 第一种方法是用现在的收益曲线来贴现未来现金流并保持抵押贷款再融资利率固定于现在的抵押贷款利率[10].

因为抵押贷款再融资利率是固定的，投资者通常可以规定一个合理的提前还贷率，用于估计直到债券到期时的所有未来现金流. 第二种方法称为*零波动* OAS，通过允许抵押贷款利率沿着远期利率[11]曲线上升来计算静态差价. 在这种情况下，我们需要通过提前还款模型来决定由未来再融资利率向量所隐含的未来提前还款率向量（一个提前还款计划）. 在计算出静态差价和 OAS 之后，我们可以通过计算 OAS（在假设的利率波动下）和静态差价之间的差异来计算任何 MBS 所带有的提前还款期权中隐含的成本. 即

$$\text{期权成本} = \text{静态差价} - \text{OAS} \tag{3.8}$$

因此，由于整体上来讲，在面对利率波动不确定性时，一个额度的期权成本比其 OAS 更稳定，所以对于小的市场波动，这一额度的 OAS 可以用重新计算静态差价并减去其期权成本来估算. 这一计算十分有用，因为 OAS 的计算十分麻烦，而静态差价则很容易计算[12].

十分重要的一点是 MBS 的投资者相当于持有不可赎回债券的多头头寸和看涨（提前还款）期权的空头头寸[13]. 不可赎回债券是一组零息票债券的集合，比如零息票国库券（treasury strips)，而看涨期权赋予借款人在到期之前任何时间还款的权力[14]. 因此，MBS 的价值为不可赎回债券与看涨（提前还款）期权的价差. OAS 是 MBS 的债券部分与期权部分的差价. 计算 OAS 的两个主要输入为关于本金（计划和非计划的）和息票支付的现金流函数，以及在假设现金流贴现的零息票曲线期限结构下生成的利率路线[15]. 在每个现金流日期前，即期利率（从利率路线的期限结构的相应时点上观测）为每个现金流决定贴现因子[16].

z_t 表示期限为 t（如 t 年或 t 月）的零时刻的贴现率，把今天看做是零时刻（类似地，f_{tj} 表示 j 时刻的 t 期远期贴现率），K 表示期权调整价差. 在我们简单的 4 步二项式例子中，零时刻上的一年期远期利率 $f_{10}=8.0\%$；第 1 步的一年期远期利率是 $f_{11}=\{8.6\%, 7.45\%\}$；第 2 步的一年期远期利率为 $f_{12}=\{9.26\%, 8\%, 6.96\%\}$；第 3 步的一年期远期利率是 $f_{13}=\{9.986\%, 8.6\%, 7.45\%, 6.51\%\}$.

每条路线的价值都是通过风险调整的零时刻即期利率 z 对每个现金流进行贴现得到的. 在我们的 4 步例子中，路线 i 上的 MBS 价值是：

$$V_i = \frac{CF_1}{1+z_1} + \frac{CF_2}{(1+z_2)^2} + \frac{CF_3}{(1+z_3)^3} + \frac{CF_4}{(1+z_4)^4}$$

其中，由于可以用远期利率来表示零时刻的利率，我们得到：

$$\begin{aligned}
z_1 &= f_{10} \\
z_2 &= ((1+f_{10})(1+f_{11}))^{1/2} - 1 \\
z_3 &= ((1+f_{10})(1+f_{11})(1+f_{12}))^{1/3} - 1 \\
z_4 &= ((1+f_{10})(1+f_{11})(1+f_{12})(1+f_{13}))^{1/4} - 1
\end{aligned}$$

一般来说，对于路线 $i=1, \cdots, N$，

$$z_t = \{(1+f_{10})(1+f_{11})(1+f_{12})\cdots(1+f_{12T})\}^{1/T} - 1$$

请注意在我们的例子中，我们假定的是一年期的远期利率，但是在实际更复杂的应用中，我们可能要模拟 360 个月的未来 1 个月的远期利率. 从而，对于每一条路线，我们都可以模拟得到

360 个 1 月期而不是 4 月期的远期利率、抵押贷款再融资利率和现金流，而且可以模仿更多的路线，比如 1 024 条而非 8 条.

路线 1 的零时刻的利率计算如下：

$$z_1 = 0.08$$

$$z_2 = ((1.08)(1.074\,546))^{1/2} - 1 = 0.077\,269$$

$$z_3 = ((1.08)(1.074\,546)(1.069\,588))^{1/3} - 1 = 0.074\,703$$

$$z_4 = ((1.08)(1.074\,546)(1.069\,588)(1.065\,08))^{1/4} - 1 = 0.072\,289$$

所以路线 1 的 MBS 的价值为：

$$V_1 = \frac{\$\,335\,224}{1.08} + \frac{\$\,393\,278}{(1.077\,269)^2} + \frac{\$\,220\,507}{(1.047\,03)^3} + \frac{\$\,241\,337}{(1.072\,289)^4} = \$\,1\,009\,470$$

表 3.6 显示了 8 条路线的 MBS 计算过程.

表 3.6

路线 1	7	8	9	10	11
年	*CF*	*Z*1，*t*−1	*Zt*0	价值	概率
1	335 223.6	0.08	0.08	310 392.2	0.5
2	393 277.6	0.074 546	0.077 27	338 883.5	0.5
3	220 507.5	0.069 588	0.074 703	177 647.1	0.5
4	241 336.8	0.065 08	0.072 289	182 547.5	
			价值＝	1009 470	0.125
路线 2	**7**	**8**	**9**	**10**	**11**
年	*CF*	*Z*1，*t*−1	*Zt*0	价值	概率
1	335 223.6	0.08	0.08	310 392.2	0.5
2	393 277.6	0.074 546	0.077 27	338 883.5	0.5
3	146 020.8	0.069 588	0.074 703	117 638.5	0.5
4	321 782.4	0.074 546	0.074 664	241 252.6	
			价值＝	1 008 167	0.125
路线 3	**7**	**8**	**9**	**10**	**11**
年	*CF*	*Z*1，*t*−1	*Zt*0	价值	概率
1	335 223.6	0.08	0.08	310 392.2	0.5
2	153 480.3	0.074 546	0.077 27	132 252.5	0.5
3	231 199.6	0.08	0.078 179	184 465.3	0.5
4	509 488.7	0.074 546	0.077 27	378 300.6	
			价值＝	1 005 411	0.125
路线 4	**7**	**8**	**9**	**10**	**11**
年	*CF*	*Z*1，*t*−1	*Zt*0	价值	概率
1	195 578	0.08	0.08	181 090.8	0.5
2	182 257.9	0.086	0.082 996	155 393.5	0.5
3	274 549.5	0.08	0.081 996	216 742.2	0.5
4	605 017.9	0.074 546	0.080 129	444 493.9	
			价值＝	997 720.3	0.125
路线 5	**7**	**8**	**9**	**10**	**11**
年	*CF*	*Z*1，*t*−1	*Zt*0	价值	概率
1	335 223.6	0.08	0.08	310 392.2	0.5
2	153 480.3	0.074 546	0.077 27	132 252.5	0.5
3	142 746.7	0.08	0.078 179	113 892.1	0.5

（续）

路线 5 年	7 CF	8 Z1，t−1	9 Zt0	10 价值	11 概率
4	605 017.9	0.086	0.801 29	444 493.9	
			价值＝	1001 031	0.125
路线 6 年	7 CF	8 Z1，t−1	9 Zt0	10 价值	11 概率
1	195 578	0.08	0.08	181 090.8	0.5
2	182 257.9	0.086	0.082 996	155 393.5	0.5
3	169 511.7	0.08	0.081 996	133 820.4	0.5
4	718 458.7	0.086	0.082 996	522 269.5	
			价值＝	9 925 741	0.125
路线 7 年	7 CF	8 Z1，t−1	9 Zt0	10 价值	11 概率
1	195 578	0.08	0.08	181 090.8	0.5
2	182 257.9	0.086	0.082 996	155 393.5	0.5
3	169511.7	0.092 6	0.086 188	132 277.2	0.5
4	718 458.7	0.086	0.086 141	516 246.6	
			价值＝	985 008	0.125
路线 8 年	7 CF	8 Z1，t−1	9 Zt0	10 价值	11 概率
1	195 578	0.08	0.08	181 090.8	0.5
2	182 257.9	0.086	0.082 996	155 393.5	0.5
3	169 511.7	0.092 6	0.086 188	132 277.2	0.5
4	718 458.7	0.099 86	0.089 59	509 741.1	
			价值＝	978 502.5	0.125

资料来源：Johnson S(2004)．

最后一步是通过 8 条路线的平均价值来计算 MBS 的理论价值．在本例中，抵押贷款组合的理论价值是 8 条路线的 MBS 价值的平均值：

$$\overline{V} = \frac{1}{N}\sum_{i=1}^{N} V_i \tag{3.9}$$

根据公式（3.9），理论价值为＄997 235 或票面价值的 99.723 5％．请注意，除理论价值外，我们还可以确定分布方差：

$$\mathrm{Var}(V) = \frac{1}{N}\sum_{i=1}^{N}[V_i - \overline{V}]^2 \tag{3.10}$$

同样，我们还可以通过加权平均每一条路线的 MBS 价值来计算理论价值，其权重是路线上每一价值的概率（每一个分叉点上升和下降的概率都为 0.5），见表 3.7.

表 3.7

价 值	概 率	价 值	概 率
1 009 470	0.125	992 574.1	0.125
1 008 167	0.125	985 008	0.125
1 005 411	0.125	978 502.5	0.125
997 720.3	0.125	加权平均价值	＄997 235
1 001 031	0.125		

3.5 在 C++ 中应用蒙特卡罗方法为 MBS 定价

为了在 C++ 中给 MBS 定价，我们创造并且定义一个 MBS 类别，包含通过对二叉树的即期利率路线进行**蒙特卡罗模**拟来对 MBS 定价的方法.

```
#ifndef _MBS_H__
#define _MBS_H__

#include <vector>
#include "math.h"
#include "time.h"
#include "Utility.h"
#include "TNT.h"
#define SIZE_X 100
#define SIZE_Y 100

using namespace std;
```

我们定义两个全局二维数组变量，用于存储二叉树的即期利率和贴现率.

```
static TNT::Array2D<double> spotRate(SIZE_X,SIZE_Y);
static TNT::Array2D<double> discountRate(SIZE_X,SIZE_Y);
```

这个 MBS 类别包含一个重载构造函数，可以将名义本金、息票、加权平均 WAC、加权平均到期日（WAM）以及期权调整价差（OAS）考虑进去. 这个类别包含 calcPrice 方法，这种方法先构建一棵二叉树，然后对二叉树上的利率路线进行蒙特卡罗模拟. calcPrice 方法将初始即期利率、抵押贷款再融资利率、二叉树的步骤数量和模拟次数考虑进去. MBS 类还包含一种计算条件提前还款率（CPR）的方法 calcCPR，它考虑了目前的再融资利率和根据二叉树现在的时间点与现在路线上的贴现率存储档案来计算现行贴现率的方法 computeZeroRates. 这个 MBS 类别包含一个 calcPayment 函数，这个函数将剩余本金和到期日作为输入变量来计算现在的抵押货款还款额. 最后，这个 MBS 类别包含 getPrice（计算得到 MBS 的价格）、getStdDev 方法（计算得到 MBS 价格的标准差）以及 getStdError 方法（计算得到 MBS 价格的标准误差）.

MBS.h

```
// MBS.h: interface for the MBS class.
//
//////////////////////////////////////////////////////////////////////

#ifndef _MBS_H__
#define _MBS_H__

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include <vector>
#include "math.h"
#include "time.h"
#include "Utility.h"
#include "TNT\TNT.h"
#define SIZE_X 100
#define SIZE_Y 100
```

```
using namespace std;
static TNT::Array2D<double> spotRate(SIZE_X,SIZE_Y);
static TNT::Array2D<double> discountRate(SIZE_X,SIZE_Y);

class MBS
{
  public:
        MBS();
        MBS(double principal, double coupon, double WAC, double WAM,
          double OAS) :
                faceValue(principal), coupon(coupon), WAC(WAC), WAM(WAM),
                OAS(OAS), T(WAM) { }
        virtual ~MBS()  { }
        double calcPayment(double principal, double T);   // compute
                                                          // payment
                                                          // amount
        void calcPrice(double initRate, double financeRate, int N,
          long int M);
        double calcCPR(double rate);
        void buildTree(double initRate, double financeRate, int N);
        double computeZeroRates(int cnt, vector<double> rate);
        double calcSMM(double x);
        double getPrice();
        double getStdDev();
        double getStdErr();
        double getMaturity();
        double getWAM();
        double getWAC();
        double getOAS();
  private:
        double OAS;               // option adjusted spread
        double faceValue;         // principal amount
        double coupon;            // coupon rate
        double WAM;               // weighted average maturity
        double WAC;               // weighted average coupon
        vector<double> zeroRates; // store discount zero coupon rates
        double T;                 // maturity of MBS
        double mbsPrice;          // price
        double stdDev;            // standard deviation
        double stdErr;            // standard error
};

#endif _MBS_H__
```

这个算法的定义是：

MBS.cpp

```
// MBS.cpp: implementation of the MBS class.
//
//////////////////////////////////////////////////////////////////////

#include "MBS.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
```

```
void MBS::buildTree(double initRate, double financeRate, int N)
{

        Utility util;
        double u = 1.1;
        double d = 1/u;
        double p = (exp(initRate*T) - d)/(u - d);
        double deviate = 0.0;
        long seed = 0;
        double refRate = financeRate;
        long* idum = 0;
        double pay = faceValue;
        double faceAmount = 0.0;
        double interest = 0.0;
        double schedulePrincipal = 0.0;
        double prepaidPrincipal = 0.0;
        double CPR = 0.0;
        double balance = faceValue;
        double sum = 0.0;
        double totalsum = 0.0;
        double SMM = 0.0;
        TNT::Array1D<double> CF(SIZE_X);   // cash_flow
        vector<double> disc(0.0);

        srand(unsigned(time(0)));
        seed = (long) rand() % 100;
        idum = &seed;
        // build binomial tree for rates
        for (int i = 0; i <= N; i++)
        {
                for (int j = 0; j <= i; j++)
                {
                        spotRate[i][j] = initRate*pow(u,j)*pow(d,i-j);
                        discountRate[i][j] = spotRate[i][j] + OAS;
                }
        }


        faceAmount = faceValue;
        int k = 0;
        long int M = 10000;
        int cnt = 0;
        double r = 0.0;
        int j = 0;

        for (k = 0; k < M; k++)
        {
                sum = 0.0;
                balance = faceValue;
                refRate = financeRate;
                j = 0;
                disc.clear();
                disc.empty();
                disc.push_back(discountRate[0][0]);

                for (i = 0; i < N; i++)
                {
                        balance = balance - (schedulePrincipal +
                                        prepaidPrincipal);
                        deviate = util.gasdev(idum);
```

```
                        if (deviate > 0)
                        {
                                j++;
                                refRate = refRate*u;
                        }
                        else
                        {
                                j--;
                                if (j < 0)
                                        j = 0;
                                refRate = refRate*d;
                        }
                        disc.push_back(discountRate[i+1][j]);
                        interest = coupon*balance;
                        pay = calcPayment(balance,WAM-i);
                        schedulePrincipal = pay - interest;

                        if (balance >= schedulePrincipal)
                        {
                                CPR = calcCPR(refRate);
                                SMM = calcSMM(CPR);
                                prepaidPrincipal = SMM*(balance -
                                  schedulePrincipal);

                                if (i != N-1)
                                        CF[i] = interest +
                                                  schedulePrincipal +
                                                  prepaidPrincipal;
                                else
                                        CF[i] = interest + balance;

                                r = computeZeroRates(i,disc);
                                sum = sum + CF[i]/(pow(1+r,i+1));

                        }
                        else
                                goto x;

                }
                x:
                totalsum = totalsum + sum;
        }
        double ave = (totalsum/M);
        std::cout << "MBS price = " << ave << endl;
}

double MBS::calcCPR(double rate)
{
        double CPR = 0.0;
        double value = WAC - rate;

        /*
        if (value <= 0)
                CPR = 0.05;
        else if ((value <= 0.005) && (value > 0))
                CPR = 0.10;
        else if ((value <= 0.01) && (value > 0.005))
                CPR = 0.20;
        else if ((value <= 0.0125) && (value > 0 01))
```

```
                CPR = 0.30;
        else if ((value <= 0.02) && (value > 0.0125))
                CPR = 0.40;
        else if ((value <= 0.025) && (value > 0.02))
                CPR = 0.50;
        else if ((value <= 0.03) && (value > 0.025))
                CPR = 0.60;
        else
                CPR = 0.70;
        */

        CPR = 100*(1-pow((1-(value/100)),12));

        return CPR;

}

double MBS::calcPayment(double fv, double T) {
        return (fv*coupon)/(1-pow(1/(1+coupon),T));
}

void MBS::calcPrice(double initRate, double financeRate, int N,
  long int M){
       Utility util;         // utility class for generating
                             // random deviates
       double u = 1.1;       // up move in binomial tree
       double d = 1/u;       // down move in binomial tree
       double p = (exp(initRate*T) - d)/(u - d);  // up probablity
       double deviate = 0.0;                      // random deviate
       long seed = 0;                             // seed
       double refRate = financeRate;    // refinance rate
       long* idum = NULL;               // pointer to seed value for RNG
       double pay = faceValue;          // face value of MBS
       double faceAmount = 0.0;         // face amount
       double interest = 0.0;           // interest payment
       double schedulePrincipal = 0.0;  // scheduled principal payments
       double prepaidPrincipal = 0.0;   // prepaid principal payments
       double CPR = 0.0;                // conditional prepayments
       double SMM = 0.0;                // monthly mortality
       double balance = faceValue;      // balance remaining
       double sum = 0.0;                // sum of discounted cash flows
                                        // along a path
       double totalsum = 0.0;  // total sum of all discounted cash flows
       double totalsum2 = 0.0;
       TNT::Array1D<double> CF(SIZE_X); // cash_flow
       vector<double> disc(0.0);        // stores discount rates

       // build binomial tree for rates
       for (int i = 0; i <= N; i++)
       {
               for (int j = 0; j <= i; j++)
               {
                       spotRate[i][j] = initRate*pow(u,j)*pow(d,i-j);
                       discountRate[i][j] = spotRate[i][j] + OAS;
               }
       }

       srand(unsigned(time(0)));
       seed = (long) rand() % 100;
       idum = &seed;
```

```
faceAmount = faceValue;
int k = 0;
int cnt = 0;
double r = 0.0;
int j = 0;

for (k = 0; k < M; k++)
{
        sum = 0.0;
        balance = faceValue;
        refRate = financeRate;
        j = 0;
        disc.clear();
        disc.push_back(discountRate[0][0]);

        for (i = 0; i < N; i++)
        {
                balance = balance - (schedulePrincipal +
                            prepaidPrincipal);
                deviate = util.gasdev(idum);
                if (deviate > 0)
                {
                       j++;
                       refRate = refRate*u;
                }
                else
                {
                       j--;
                       if (j < 0)
                        j = 0;
                       refRate = refRate*d;
                }
                disc.push_back(discountRate[i+1][j]);
                interest = coupon*balance;
                pay = calcPayment(balance,WAM-i);
                schedulePrincipal = pay - interest;

                if (balance >= schedulePrincipal)
                {
                   CPR = calcCPR(refRate);
                   SMM = calcSMM(CPR);
                     prepaidPrincipal = SMM*(balance -
                                         schedulePrincipal);

                     if (i != N-1)
                        CF[i] = interest + schedulePrincipal +
                                  prepaidPrincipal;
                     else
                        CF[i] = interest + balance;

                    r = computeZeroRates(i,disc);
                    sum = sum + CF[i]/(pow(1+r,i+1));

                }
                else   // break out of loop
                   goto x;

        }
```

```
                x:
                  totalsum = totalsum + sum;
                  totalsum2 = totalsum2 + sum*sum;
        }
        double ave = (totalsum/M);

        mbsPrice = ave;
        stdDev = sqrt(totalsum2 - (double)(totalsum*totalsum)/M)*(exp(-
        2*initRate*T)/(M-1));
        stdErr = (double) stdDev/sqrt(M);
}


double MBS::calcSMM(double CPR) {
         return (1 - pow((1 - CPR),(double)1/12));
}

double MBS::computeZeroRates(int cnt, vector<double> rate) {
        double value = WAC+1;
        for (int j = 1; j <= cnt; j++)
            value = value*(1 + rate[j]);

        if (cnt == 0)
            value = WAC;
        else
           value = pow(value,(double)1/(cnt+1)) - 1;

        return value;
}

double MBS::getPrice() {
     return mbsPrice;
}

double MBS::getStdDev() {
     return stdDev;
}

double MBS::getStdErr() {
     return stdErr;
}

double MBS::getMaturity() {
        return T;
}

double MBS::getWAM() {
        return WAM;
}

double MBS::getWAC() {
        return WAC;
}

double MBS::getOAS() {
        return OAS;
}
```

考虑用之前用过的参数来为 MBS 定价：

Main.cpp

```
#include <fstream.h>
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include <math.h>
#include <map>

#define SIZE_X 100
#include "CMO.h"

void main()
{
        std::cout.precision(7);
        double principal = 1000000;      //  underlying principal
                                         //  (notional) of MBS
        double coupon = 0.08;            //  coupon rate
        double WAC = 0.08;               //  weighted average
                                         //  coupon rate
        double WAM = 10;                 //  weighted average maturity
        double OAS = 0.02;               //  option adjusted spread
        double initSpotRate = 0.06;      //  spot rate
        double initRefinanceRate = 0.08; //  refinance rate
        int N = 10;                      //  number of time steps in tree
        long int M = 100000;             //  number of simulation paths
        MBS mbs(principal,coupon,WAC,WAM,OAS);

        std::cout << "Running Monte Carlo to price MBS..." << endl << endl;
        mbs.calcPrice(initSpotRate,initRefinanceRate,N,M);
        std::cout << "MBS Price = " << mbs.getPrice() << endl;
        std::cout << "Std Deviation = " << mbs.getStdDev() << endl;
        std::cout << "Std Error = " << mbs.getStdErr() << endl << endl;

        std::cout << "Pricing MBS with Simulations of Binomial
          Tree Paths..." << endl;
        MBS mbs1(principal,coupon,WAC,N,OAS);
        mbs1.buildTree(initSpotRate,coupon,N);

        vector<Tranche> tranche;
        Tranche trA('A',500000,0.06);
        tranche.push_back(trA);
        Tranche trB('B',300000,0.065);
        tranche.push_back(trB);
        Tranche trC('C',200000,0.07);
        tranche.push_back(trC);
        Tranche trZ('Z',100000, 0.075);
        tranche.push_back(trZ);

        std::cout << endl;
        std::cout << "Pricing CMO Tranches..." << endl << endl;
        CMO cmo(mbs,tranche);
        cmo.calcCashFlows(initSpotRate,initRefinanceRate,N,M);
}
```

结果如下：

```
MBS Price = 964386.69
Std Deviation = 110.07
Std Error = 1.10
```

我们可以通过增加模拟次数来提高精确度. 例如，如果 $M=100\,000$，则

```
MBS Price = 964469.78
Std Deviation = 34.86
Std Error = 0.11
```

因此，MBS 的价格大约等于面值的 96.5%. 时间步数越多，计算精度越高.

连续时间模型

二叉树模型是一个简单的离散模型，不能体现利率的实际波动，因为在每个时间点上，利率只能上升或下降，不能保持不变或者在两个时间点之间波动. 为了表示利率波动的实际过程，我们通常使用利率期限结构的无套利模型. 短期利率假定遵循扩散（连续时间随机）过程. 用短期利率的变化来表示这些模型的一般形式如下：

$$dr_t = k(\theta - r)dt + \sigma r^{\alpha} dz_t, \quad r(0) = r_0$$

其中 dr_t 表示在无穷短的时间 dt 内，r_t 无穷小的变化，dz_t 是一个标准 Wiener 过程. k 表示均值回归速度，θ 表示利率过程的长期平均值，α 表示部分有条件的波动性指数，σ 表示 r_t 变化的瞬时标准差. 大量短期利率模型的 α 参数不同. Vasicek 模型假设 $\alpha=0$，Cox-Ingersoll-Ross（CIR）模型假设 $\alpha=0.5$，而 Courtadon 模型假设 α 为 1.

为了模拟这一过程，我们对其进行离散化如下（假定 Courtadon 模型）：

$$\Delta r_t = k(\theta - r_t)\Delta t + \sigma r_t \sqrt{\Delta t z_t}, \quad r(0) = r_0$$

很多利率模型都有一些均值回归形式，将得到的利率路线回归到“长期”水平. 如果没有回归，则利率会出现不合理的过高或过低. 波动性也会逐渐在理论上趋近无穷、类似地，一个高比例的波动性假设会导致更高的收益波动，从而使得再融资的概率更大. 再融资概率的增加意味着暗含的看涨期权的价值更高，从而导致更高的期权成本[17].

除更符合实际的期限结构外，我们还需要扩展提前偿还模型来反映多种因素对提前偿还的效应. 我们可以利用 Richard 和 Roll（1989）提前偿还模型，它是基于对借款人融资条件的经验估计. 这个模型试图通过观察实际提前还款金额并将其与提前还款经济理论中的可度量因素相联系来解释提前还款. 这一提前偿还模型有几个假设. CPR 最高为 50%，最低为 0%. 当 WAC 再融资利率差异为 200 个基点时，CPR 等于中点 25%. 中点的斜率最大，利率变化 10 个基点，CPR 变化 6%.

Richard 和 Roll（1989）模型确定了任何提前偿还模型都包含的 4 个因素[18]：

1. 再融资动机：借款人再融资的动机

$$RI(t) = a + b(\arctan(c + d(\text{WAC} - r_t)))$$

其中[19]

$$a = (\max \text{CPR} + \min \text{CPR})/2$$
$$V = 100(\max \text{CPR} - a)/(\pi/2)$$
$$d = \max \text{斜率} / b$$
$$c = -dx \text{ 中点差值}$$

2. 时间调整（抵押贷款期限）

$$Age(t) = \min\left(\frac{t}{30}, 1\right)$$

3. 季节性（月度乘数）：住房营业额的年趋势[20]

$$MM(t) = (0.94, 0.76, 0.74, 0.95, 0.98, 0.92, 0.98, 1.10, 1.18, 1.22, 1.23, 0.98)$$

其中 t 表示第 t 个月，$t=1$，…，12.

4. 倦怠乘数：再融资高峰之后的逐渐消减

$$BM(t) = 0.3 + 0.7\frac{B(t)}{B(0)}$$

其中 $B(t)$ 是 t 时刻的抵押贷款余额，$B(0)$ 是初始抵押贷款金额.

年度提前还款率 $\mathrm{CPR}(t)$ 为：

$$\mathrm{CPR}(t) = RI(t)x\,Age(t)x\,MM(t)x\,BM(t)$$

在这个扩展的提前还款模型中，MBS 的现金流为：

- $MP(t)$ 是 t 期（月）的抵押贷款计划偿还金额：

$$MP(t) = B(t)\left(\frac{WAC/12}{1-(1+WAC/12)^{-WAM+t}}\right)$$

- $IP(t)$ 为 t 期的利息支付：

$$IP(t) = B(t)\left(\frac{WAC}{12}\right)$$

- $PP(t)$ 为 t 期的提前偿还本金金额：

$$PP(t) = \mathrm{SMM}(t)(B(t) - SP(t))$$

其中

$$\mathrm{SMM}(t) = 1 - \sqrt[12]{1-\mathrm{CPR}(t)}, \quad SP(t) = MP(t) - IP(t)$$

- $SP(t)$ 是 t 期的计划偿还本金金额，$\mathrm{SMM}(t)$ 是 t 时刻的月度坏账率；

每个月抵押贷款余额减少额为：

$$B(t+1) = B(t) - TPP(t)$$

其中 $TPP(t)$ 是 t 期的全部本金偿还额. 如同之前一样，可以用这些公式和蒙特卡罗模拟来计算 MBS t 时刻的预期现金流 $CF(t)$，从而计算 MBS 的价格 P：

$$P = E^Q\left[\sum_{t=0}^{M} PV(t)\right] = E^Q\left[\sum_{t=0}^{M} \mathrm{d}f(t)CF(t)\right]$$

其中

$$PV = t \text{ 时刻现金流的现值}$$

$$\mathrm{d}f(t) = \prod_{k=1}^{t}\frac{1}{(1+r_k)} = t \text{ 时刻的贴现因子}$$

$$CF(t) = MP(t) + PP(t) = TTP(t) + IP(t)$$

$$MP(t) = SP(t) + IP(t)$$

$$TPP(t) = SP(t) + PP(t)$$

蒙特卡罗模拟可以用来帮助人们（如基金经理）判断 MBS 的当前市场价格与理论价值和方差相比，是过高还是过低，从而抓住偏离真实价值的“错误定价”的机会进行交易获得利润. 这个错误定价是市场价和“真实的”理论值之间的差. 人们可以利用模拟得到的信息来估计每条路线的平均寿命和所有路线的均值与方差. 我们可以据此估计提前偿还风险.

现金流分析方法有两种. 第一种是静态现金流分析，假定 PSA 为一个固定值；第二种是向量（或动态）现金流分析，假定 PSA 随时间变化而变化. 静态现金流分析方法基于不同的 PSA 速率来估计 *CF*，然后根据价格和不同的 PSA 速率来计算 *CF* 的收益，假设利率波动性固定. 在已知基于 PSA 速率、久期、平均寿命和其他抵押贷款或 MBS 的特性的收益估计值的情况下决定最优价格时，静态现金流分析有用. 表 3.8 展示了对于不同面值、价格和 PSA 速率假设的静态分析.

表 3.8

面　值	PSA 收　益			均　值	标准差
（价格以面值的%表示）	50%	100%	165%		
$44.127M (90.75)	8.37%	9.01%	9.76%	9.047%	0.568 1
$45.100M (92.75)	7.82%	8.31%	8.88%	8.336 7%	0.433 0
$46.072M (94.75)	7.29%	7.63%	8.03%	7.650 0%	0.523 4
$47.045M (96.75)	6.78%	6.97%	7.20%	6.980 0%	0.171 7
$48.017M (98.75)	6.28%	6.34%	6.40%	6.340 0%	0.049 0
平均寿命（年）	5.10	3.80	2.93		
到期期限	9.40	7.15	5.40		
久期	4.12	3.22	2.57		
向量分析					
月：	PSA				
1～36	50	100	165		
37～138	200	200	400		
139～357	300	300	400		
要求回报为 $48.127M：					
收益	6.02%	6.01%	6.00%	6.010 0%	0.008 16
平均寿命	3.51	2.71	2.63		
久期	2.97	2.40	2.34		

资料来源：Johnson S(2004).

基于 *CF* 分析，如果投资者想从一个 SPA 为 165（或者等同于一个平均寿命为 2.93、久期为 2.57 的投资）的 MBS 投资中得到 9.76% 回报，则投资者最多愿意支付 MBS 面值的 90.75%.

向量分析是一种更加动态的方法. 向量分析可以像静态现金流分析一样用来决定要求

收益下的价格. 表 3.8 下半部分就展示了对于 3 个子时间段的不同 PSA 速率假设的向量分析.

通常来说，PSA 下降对较长期限资产的益处要大于对较短期限资产的益处. 放慢提前偿还速度会增加所有证券的 OAS，进而会使溢价交易的证券增加更多，同时也会提高证券价格. 但是，对于久期较短的证券 OAS 变化带来的价格变化没有久期较长的证券大. 相反，PSA 的增加会降低所有证券的价格和 OAS，尤其是溢价交易时. PSA 增加对纯利息证券（IO）和纯利息类证券的影响相反. 利率波动性的降低会增加所有证券的 OAS 和价格，尽管这种增加大部分由较长期限的证券来实现. 每种证券的 OAS 收益几乎与那些证券的 OAS 久期一致[21]. 利率波动性的增加会分配抵押品的损失，使得证券久期越长，损失越大.

作为估价模型的一部分，期权调整久期和期权调整凸性是十分重要的度量. 通常来说，久期衡量债券价格对于很小的利率变化的敏感性. 久期可以理解为收益曲线平移 100 个基点，价格变化的近似百分比[22].

举个例子，如果一个债券的久期为 3.4，这表明利率上升 100 基点会导致价格下降近 3.4%. 收益增加 50 个基点会导致价格下降近 1.7%. 基点变化越小，对变化的估计越精确.

MBS（或者任何固定收益证券）的有效久期近似为：

$$\text{有效久期} = \frac{V_{-} - V_{+}}{2V_0 \Delta r} \tag{3.11}$$

其中

V_{-} = 收益下降（每 \$100 面值）Δr 时的价格

V_{+} = 收益上升（每 \$100 面值）Δr 时的价格

V_0 = 初始价格（每 \$100 面值）

Δr = 用于计算 V_{-} 和 V_{+} 的利率的基点变化数

与修正久期（久期的标准衡量）相比，有效久期假定公式（3.11）的价格计算是建立在现金流随利率变化的假设基础之上的. 而修正久期假定现金流不随利率变化而变化，所以修正久期适合用来衡量无期权证券，如国债，但不适合用来衡量包含嵌入期权的证券，如 MBS，因为其现金流受利率变化的影响. 因此，MBS 使用有效久期，也被称为 OAS 久期，可用以下的 OAS 模型计算得到. 首先，利用现行的利率期限结构找到债券的 OAS. 然后保持 OAS 恒定，移动期限结构两次——一次增加收益，一次减少收益，分别得到两个价格 V_{-} 和 V_{+}[23].

因此有效久期可以结合二叉树或 *CF* 分析来衡量具有期权风险的债券或 MBS 的久期. 利用二叉树对包含嵌入期权的债券估价有以下几个步骤. 首先，利用自举法估计收益曲线并利用校准方法对债券估价 V_0. 然后利用自举法使估计收益曲线下降一个很小的数量，利用校准方法来估计债券价格 V_{-}. 之后利用自举法使估计收益曲线上升一个很小的数量，利用校准方法来估计债券价格 V_{+}. 最后，用公式（3.11）计算有效久期. 类似地，也可以利用静态现金流分析计算有效久期. 给定 PSA，确定与小的收益变化相关的价格（也可以应用 PSA 随利率变化的模型），然后运用公式（3.11）. 值得注意的是，有效久期仅仅假设期限结构的平行移动，如果移动不是平行的，则不能准确预计债券的价格变化.

凸性（convexity）衡量的是无法用久期解释的证券价格变化. 它可以被看做是债券价格关

于收益函数的泰勒展开形式的二阶项. 如果收益变化一定基点，具有正凸性的债券价格上升的百分比要大于下降的百分比. 相反，如果收益变化一定基点，具有负凸性的债券价格下降的百分比要大于上升的百分比. 尽管正凸性是债券的一个很好的特性，但证券可以是正凸性也可以是负凸性，这取决于现行的抵押贷款再融资利率与标的抵押贷款利率的对比. 凸性可以这样计算：

$$\frac{V_{+}+V_{-}-2(V_0)}{2V_0(\Delta r)^2} \tag{3.12}$$

如果现金流不随收益变化，那么公式（3.12）中得到的凸性是对一个无期权债券的标准凸性的很好近似. 但是，如果公式（3.12）中得到的价格是从收益变化带来的现金流变化（通过改变提前偿还率）中导出的，则得到的凸性是有效凸性（effective convexity）[24]. 如果价格是通过蒙特卡罗模拟 OAS 或 OAS 模型得到的，那么得到的值称为 OAS 凸性.

作为一个计算久期和凸性的例子，考虑一个 PSA165 MBS，价格和收益如表 3.9 所示.

表 3.9

价格	收益
102.187 5	6.75%
100.281 3	7.00%
98.406 3	7.25%

根据公式（3.12），可得久期为：

$$\frac{102.1875-98.4063}{2\ (100.2813)\ (0.0025)}=7.54$$

凸性为：

$$\frac{102.1875+98.4063-2(100.2813)}{2(100.2813)(0.0025)^2}=24$$

因此，对于收益变化 25%，债券价格变化 7.54%，具有正凸性 24%，意味着有 24%的价格变化不能用久期解释.

图 3.4 显示息票利率为 8.5%、初始价值为 100 万美元的 30 年期 MBS 的现金流模拟.

MBS 定价模型的基本因素遵循扩散过程，单因素（利率）估价模型见 Kariya 和 Kobayashi（2000），三因素（利率、抵押贷款利率和房地产价格）估价模型见 Kariya、Ushiyama 和 Pliska（2002）[25].

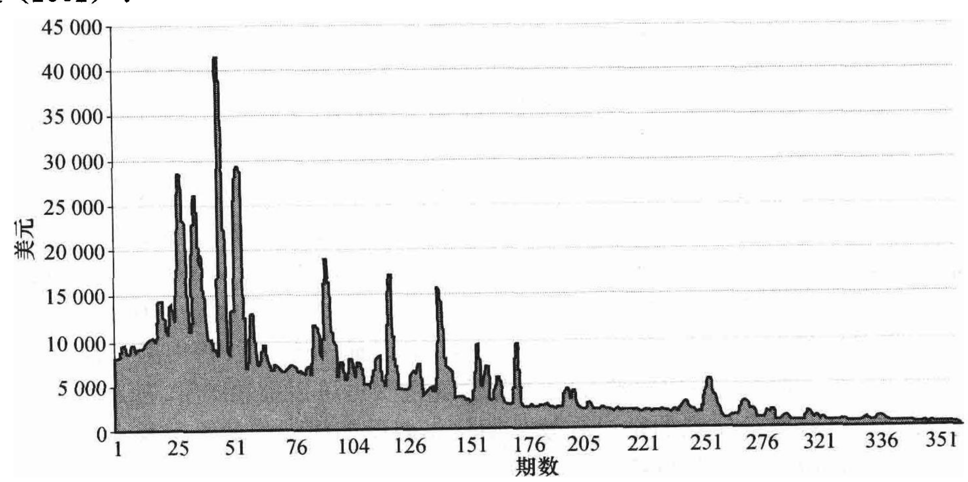


图 3.4 30 年期 MBS 的现金流模拟

3.6 使用 Matlab 固定收益工具包对 MBS 估价

Matlab 固定收益工具包可以用来对 MBS 估价并计算很多 MBS 度量，比如有效久期、凸性和 OAS. 以下变量是用 Matlab 对 MBS 估价的输入项：

- **Price**：每 100 美元面值的净价.
- **Yield**：抵押贷款收益，每月复利（十进制）.
- **Settle**：交割日期. 一系列日期或日期列. 必须在到期日或到期日之前进行交割.
- **OriginalBalance**：初始资产价值，以美元计价（每期期初的资产余额）.
- **TermRemaining**：（可选）交割日期和到期日之间的完整月份数量.
- **Maturity**：到期日. 一系列日期数或日期列.
- **IssueDate**：发行日. 一系列日期数或日期列.
- **GrossRate**：毛息票利率（包括费用），十进制. 等同于 WAC.
- **CouponRate**：净息票利率，十进制. 默认为 GrossRate. 等于 PT 率.
- **Delay**：（可选）房屋业主付款到债券持有者收到利息之间的延迟时间（按天计）. 默认为 0（没有延迟）.
- **NMBS**：住房抵押贷款证券的数量.
- **PrepaySpeed**：（可选）条件提前还款率（CPR）和基准模型之间的关系. 默认为 0. 如果你要输入一个自定的提前还款矩阵，则设 PrepaySpeed 为 [].
- **PrepayMatrix**：（可选）只有在 PrepayModel 和 PrepaySpeed 未被指定时才应用. 自定提前还款向量：一个大小为 max(TermRemaining)×NMBS 的 NaN 填充矩阵. 每列对应每个 MBS，每行对应交割后的每月.
- **ZeroMatrix**：一个三列矩阵. 第一列：一系列日期的数量. 第二列：到期日相对于第一列的即期利率. 第三列：第一列中利率的复合. 数值是 1（每年）、2（每半年）、3（每 4 个月）、4（每季度）、6（每两个月）、12（每月）和 −1（连续的）.
- **Interpolation**：插入算法. 计算相对于债券的现金流的即期利率. 可用算法为：（0）最近的，（1）线性的，还有（2）立方的. 默认值为 1.

所有这些输入项（除了 PrepayMatrix 和 ZeroMatrix）都是 NMBS×1 的向量. 以下变量都是在为债券定价时使用的输入项，可以用其找到对住房抵押贷款证券定价的收益曲线：

- **Face**：（可选）组合中每一个债券的面值. 默认值为 100.
- **Yield**：含有金融工具到期收益率的标量或向量.
- **Settle**：交割日期. 一系列日期数的标量或向量. 必须在到期日之前或等同于到期日.
- **Maturity**：到期日. 一系列日期数或日期列的标量或向量.
- **ConvDates**：债券的转换日期. 一系列日期数的矩阵.
- **CouponRates**：含有投资组合以小数形式表示的债券息票利率的矩阵. 矩阵第一列包含 Settle 和 ConvDates 中第一列日期之间使用的利率.
- **Period**：（可选）债券每年的付息次数. 一个整数向量，取值为 0，1，2，3，4，6 和 12. 默认值为 2（半年付息一次）.

- **Basis**：（可选）金融工具的每日结算基础. 一个整数向量. 0＝实际值/实际天数（默认值），1＝30/360，2＝实际值/360，3＝实际值/365，4＝30/360（PSA 使用），5＝30/360（ISDA 使用），6＝30/360（欧洲使用），还有 7＝实际值/365（日本使用）.
- **EndMonthRule**：（可选）月末法则. 一个向量. 这个法则只适用于到期日是月末，并且该月有 30 天或者更少天数. 0＝忽略法则，表示债券付息日总是与月份数值相同. 1＝启用法则（默认值），表示债券付息日总是一个月的最后一天.

要计算一个初始余额为＄10 000 000，PSA＝150，WAC＝8.125％，期限为 360 个月，剩余 357 个月（三个月的调整期）的 FHLMC 抵押贷款组合的现金流和余额. Matlab 代码如下：

```
OriginalBalance = 1000000;
GrossRate = 0.08125;
OriginalTerm = 360;
TermRemaining = 357;
PrepaySpeed = 125;
[Balance, Payment, Principal, Interest, Prepayment] =
  mbspassthrough(OriginalBalance,...
 GrossRate, OriginalTerm, TermRemaining, PrepaySpeed)
```

代码输出结果如表 3.10 所示（余额、偿还金额、本金、利息和提前偿还金额）.

表 3.10

月	余 额	偿还金额	本 金	利 息	提前偿还金额
1	998 490.00	7 439.7	668.837 3	6 770.8	836.6
2	996 780.00	7 433.4	672.802 1	6 760.6	1 045.4
3	994 850.00	7 425.7	676.647 9	6 749	1 253.8
4	992 700.00	7 416.3	680.371 9	6 735.9	1 461.6
5	990 350.00	7 405.4	683.971 6	6 721.4	1 668.7
6	987 790.00	7 392.9	687.444 3	6 705.5	1 875
7	985 020.00	7 378.9	690.787 6	6 688.2	2 080.4
8	982 040.00	7 363.4	693.999 1	6 669.4	2 284.7
9	978 850.00	7 346.3	697.076 3	6 649.2	2 487.7
10	975 460.00	7 327.7	700.017	6 627.7	2 689.5
350	5 640.00	832.5	788.748 7	43.8	36.7
351	4 820.00	827.1	788.946 9	38.2	31.4
352	4 000.00	821.8	789.145 1	32.6	26.1
353	3 190.00	816.4	789.343 4	27.1	20.8
355	1 590.00	805.9	789.740 1	16.2	10.3
356	790.00	800.7	789.938 5	10.7	5.2
357	0.00	795.5	790.137	5.3	0

资料来源：Johnson S(2004).

这些值的计算方法和 3.2 节一样. 给定一个住房抵押贷款证券组合，如何用 Matlab 命令 mbsprice 函数来计算净价格和累计利息（accrued interest）. 假设这个组合的回报是 7.25％，

WAC（毛息票）为 8.5%，到期日是 2034 年 1 月 10 日，发行日是 2004 年 1 月 10 日，我们要知道 5 个结算日的净价：2004 年 3 月 10 日、2004 年 5 月 17 日、2005 年 5 月 17 日、2006 年 1 月 10 日和 2006 年 6 月 10 日，每个 MBS 证券的 PT（息票）利率分别为 7.5%、7.875%、7.75%、7.95%和 8.125%. 假设还款延迟时间为 20 天：

```
% MBS.m  : compute MBS prices and accrued interest
Yield = 0.0725;
Settle = datenum(['10-Mar-2004';'17-May-2004';'17-May-2005';'10-Jan-
2006';'10-Jun-2006']);
Maturity = datenum('10-Jan-2034');
IssueDate = datenum('10-Jan-2004');
GrossCoupon = 0.085;
CouponRate = [0.075; 0.07875; 0.0775; 0.0795; 0.08125];
Delay = 20;
Speed = 150;
[Price, Accrt] = mbsprice(Yield, Settle, Maturity, IssueDate, ...
        GrossRate, CouponRate, Delay, Speed)
```

表 3.11 显示每个结算日的净价和累计利息.

假设一个抵押贷款组合，WAM 近似为 28 年，利用之前抵押贷款组合在 2004 年 3 月 10 日的价格和息票利息，计算 PSA 速率分别为 100、150 和 200 时的抵押贷款组合在 2004 年 3 月 10 日的 OAS. 在 Matlab 中，我们首先需要建立一个由债券价格（假设所有债券都是半年付息一次）和收益构成的零矩阵：

表 3.11

结算日	价格	累计利息
2004-3-10	101.093 7	0.000 0
2004-5-17	103.280 1	0.153 1
2005-5-17	102.367 7	0.150 7
2006-1-10	103.389 7	0.000 0
2006-6-10	104.300 8	0.000 0

```
Bonds = [datenum('11/21/2004')  0.045   100   2   3   1;
         datenum('02/20/2005')  0.0475  100   2   3   1;
         datenum('07/31/2007')  0.0500  100   2   3   1;
         datenum('08/15/2010')  0.0550  100   2   3   1;
         datenum('03/15/2012')  0.0575  100   2   3   1;
         datenum('02/15/2015')  0.0600  100   2   3   1;
         datenum('03/31/2020')  0.0650  100   2   3   1;
         datenum('08/15/2025')  0.0720  100   2   3   1;
         datenum('07/20/2034')  0.0850  100   2   3   1];

Yields = [0.0421; 0.0452;  0.0482; 0.0510;  0.0532;  0.0559;
0.0620; 0.0682; 0.0785];
% Since the above is Treasury data and not "selected" agency data, an
% ad-hoc method of altering the yield has been chosen for demonstration
% purposes
Yields = Yields + 0.025*(1./[1:9]');

% Get parameters from Bonds matrix
Settle = datenum('10-Mar-2004');
Maturity     = Bonds(:,1);
CouponRate   = Bonds(:,2);
Face         = Bonds(:,3);
Period       = Bonds(:,4);
Basis        = Bonds(:,5);
EndMonthRule = Bonds(:,6);
```

```
% compute bond prices
[Prices, AccruedInterest] = bndprice(Yields, CouponRate, ...
  Settle, Maturity, Period, Basis, EndMonthRule, [], [], [], [], ...
  Face);

% uses the bootstrap method to return a zero curve given a portfolio of
% coupon bonds and their prices
[ZeroRatesP, CurveDatesP] = zbtprice(Bonds, Prices, Settle);
SpotCompounding = 2*ones(size(ZeroRatesP));
ZeroMatrix = [CurveDatesP, ZeroRatesP, SpotCompounding];
Maturity = datenum('10-Jan-2034');
IssueDate = datenum('10-Jan-2004');
GrossRate = 0.085;
Delay = 20;
Interpolation = 1;
PrepaySpeed = [100 150 200];
Price = 101.0937;
CouponRate = 0.075;
Settle = datenum('10-Mar-2004');

OAS = mbsprice2oas(ZeroMatrix, Price, Settle, Maturity, ...
       IssueDate, GrossRate, CouponRate, Delay, Interpolation, ...
       PrepaySpeed)
```

OAS 结果如表 3.12 所示.

我们可以用命令 mbsdurp（给定价格下的久期）、mbsdury（给定收益下的久期）、mbsconvp（给定价格下的凸性）和 mbsconvy（给定收益下的凸性）来计算抵押贷款组合的有效久期和凸性. 例如，接着以上的例子，我们可以通过以下命令来计算 PSA 分别为 100、150、200 的假设下组合在 2004 年 3 月 10 日的年久期、修正久期和凸性：

```
% compute regular duration and modified duration
[YearDuration, ModDuration] = mbsdurp(Price, Settle, Maturity,
  IssueDate, GrossRate,   CouponRate, Delay, PrepaySpeed)

% compute convexity
Convexity = mbsconvp(Price, Settle, Maturity, IssueDate, GrossRate,
  CouponRate, Delay,PrepaySpeed)
```

表 3.13 显示 PSA 分别为 100、150、200 的久期、修正期限和凸性.

表 3.12

PSA	OAS
100	82.667 0
150	101.851 8
200	114.608 8

表 3.13

PSA	年久期	修正久期	凸性
100	7.066 9	6.814 8	82.323 0
150	6.104 8	5.888 1	62.247 6
200	5.371 2	5.181 4	48.336 8

3.7 抵押担保债券（CMO）

CMO（Collateralized Mortgage Obligation）是由抵押贷款组合、MBS、剥离 MBS 或者 CMO 所担保的证券. 它们被构造为几个债券等级；这些等级称为份额. 每一份额对本金有不同的优先求偿权. 两类常见的 CMO 为：按序求偿份额和计划摊销份额（PAC）. 在按序求偿份额中，债券级别是根据偿还本金次序来定的. 每一份额的本金按优先次序偿还：第一优先

份额的本金全部在第二优先份额之前付清，第二优先又要比之后的更早偿付，依此类推. 这一过程持续到所有份额都得到偿付. 通常，第一份额的到期收益率（YTM）是最低的，因为它们的平均寿命最短，提前偿还风险最小. 在此之后的份额都有更长的平均寿命和更高的 YTM.

很多普通结构中的最后一个份额要在所有更短到期日得到偿付之后才能收到利息. 由于与零息票债券相似，因此，这些债券可被称为累计债券或“Z 债券”. 那些用来偿还 Z 债券利息的钱往往被用来偿还到期日更短的份额的本金，缩短它们的平均寿命.

图 3.5 表示按序偿还债券和 Z 债券之间的本金和利息现金流的假设分布.

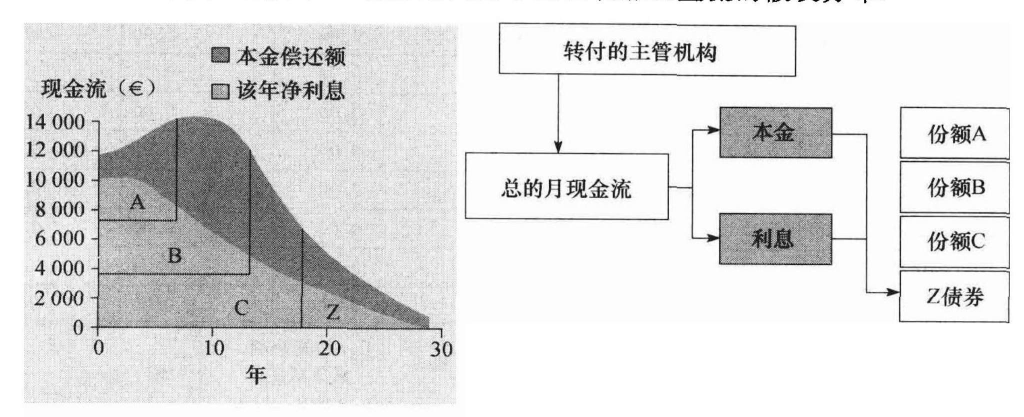


图 3.5 按序偿还债券和 Z 债券

假定我们要形成表 3.14 描述的抵押贷款组合的按序偿还份额：抵押贷款金额为 $100M，WAM = 357 个月，WAC = 8.125%，PT 率 = 7.5%，提前偿还速率=165.

表 3.14

份 额	面 值	PT 率
A	$48.625M	7.5%
B	$9M	7.5%
C	$42.375M	7.5%

这个现金流分布是这样做的. 本金偿还首先给 A，然后给 B，再给 C，然后给剩余的份额. 本金偿付包括计划的本金偿付和提前的本金偿付. 息票偿付基于份额的剩余资本. 表 3.15 显示结构化的份额偿付.

表 3.15

月	余 额 100 000 000.00	利 息	本 金	A 份额的初始余额 48 625 000.00	48 625 000.00 利 息	本 金
1	100 000 000.00	625 000.00	177 480.87	48 625 000.00	303 906.25	177 480.87
2	99 822 519.13	623 890.74	205 473.91	48 447 519.13	302 796.99	205 473.91
3	99 617 045.22	622 606.53	233 389.97	48 242 045.22	301 512.78	233 389.97
4	99 383 655.25	621 147.85	261 205.39	48 008 655.25	300 054.10	261 205.39
5	99 122 449.86	619 515.31	288 896.53	47 747 449.86	298 421.56	288 896.53
6	98 833 553.33	617 709.71	316 439.78	47 458 553.33	296 615.96	316 439.78
80	51 965 586.84	324 784.92	512 605.38	590 586.84	3 691.17	512 605.38
81	51 452 981.45	321 581.13	508 049.13	77 981.45	487.38	77 981.45

（续）

月	余　额 100 000 000.00	利　息	本　金	A份额的初始余额 48 625 000.00	48 625 000.00 利　息	本　金
82	50 944 932.32	318 405.83	503 532.52	0.00	0.00	0.00
83	50 441 399.80	315 258.75	499 055.22	0.00	0.00	0.00
84	49 942 344.58	312 139.65	494 616.89	0.00	0.00	0.00
85	49 447 727.69	309 048.30	490 217.17	0.00	0.00	0.00
99	42 968 082.27	268 550.51	432 494.82	0.00	0.00	0.00
100	42 535 587.45	265 847.42	428 635.94	0.00	0.00	0.00
101	42 106 951.51	263 168.45	424 810.66	0.00	0.00	0.00
102	41 682 140.85	260 513.38	421 018.70	0.00	0.00	0.00
103	41 261 122.15	257 882.01	417 259.77	0.00	0.00	0.00
104	40 843 862.38	255 274.14	413 533.58	0.00	0.00	0.00
105	40 430 328.80	252 689.56	409 839.84	0.00	0.00	0.00
106	40 020 488.96	250 128.06	406 178.29	0.00	0.00	0.00
356	74 797.79	467.49	37 597.30	0.00	0.00	0.00
357	37 200.49	232.50	37 200.49	0.00	0.00	0.00

B份额的 初始余额 9 000 000	9 000 000 本　金	利　息	C份额 月	C：初始余额 累计利息 42 375 000	本　金	利　息
9 000 000	0	56 250	1	42 375 000	0	0
9 000 000	0	56 250	2	42 639 843.8	0	0
9 000 000	0	56 250	3	42 904 687.5	0	0
9 000 000	0	56 250	4	43 169 531.3	0	0
9 000 000	0	56 250	5	43 434 375	0	0
9 000 000	0	56 250	54	56 411 718.8	0	0
9 000 000	196 997.83	56 250	55	56 676 562.5	0	0
8 803 002.17	899 641.259	55 018.764	56	56 941 406.3	0	0
7 903 360.91	894 022.233	49 396.006	57	57 206 250	0	0
904 465.071	850 793.615	5 652.906 7	65	59 325 000	0	0
53 671.456 51	53 671.456 5	335.446 6	66	59 589 843.8	527 084	372 436.52
0	0	0	67	59 062 759.6	575 606	369 142.25
0	0	0	68	58 487 153.2	570 502	365 544.71
0	0	0	69	57 916 651	565 442	361 979.07
0	0	0	356	74 797.793 9	37 597.3	467.486 21
0	0	0	357	37 200.493 4	37 200.5	232.503 08

资料来源：Johnson S(2004)．

为了吸引某类投资者，浮动利率和逆浮动利率的份额就产生了．这两种份额可以用一个已存在的份额（如C份额）创造出来．例如，C份额中的浮动利率级别（FR）和逆浮动利率级别

(IFR) 可以这样构造，浮动利率为 LIBOR+50 个基点，逆浮动利率为 28.5−3 (LIBOR). FR 级别占 C 份额的 75%，而 IFR 级别占 25%. 因此，份额构造如表 3.16 所示.

注意，在逆浮动利率方程中，28.5 为资本化率 (K)，3 为杠杆率 (L). 因为浮动和逆浮动利率级别都是从 C 份额中产生的，C 的利率是 7.5%. K 和 L 满足：

$$0.75[\text{LIBOR} + 0.5\%] + 0.25[K - L(\text{PLIBOR})] = 7.5\%$$

如果 LIBOR=6%，则：

$$\text{FR} = 6\% + 0.5\% = 6.5\%$$
$$\text{IFR} = 28.5 - 3(6\%) = 10.5$$
$$0.75(6.5\%) + 0.25(10.5\%) = 7.5\%$$

CMO 的份额往往利率不同. CMO 经常包括一个很特别的份额类型，称为名义纯利息 (NIO) 类型，这一类型仅收到剩余利息. 名义纯利息类型 (NIO) 常常被描述为根据名义本金支付一定利息. 以下 CMO 的 NIO 如表 3.17 所示.

表 3.16

份 额	面 值	PT 率
A	$48.625M	7.5%
B	$9M	7.5%
FR	$31.782M	LIBOR+50bps
IFR	$10.549M	28.5−3(LIBOR)

表 3.17

份 额	面 值	PT 率
A	$48.625M	6.0%
B	$9M	6.5%
Z	$42.375M	7.0%
NIO	$13.750M	7.5%

NIO 级别的名义本金为 $13.75M. 纯利息 (IO) 类型比其他级别的利息率高 7.5%. 例如，A 等级债券，IO 类型能够收到 $48.625M 的 1.5% (7.5%−6.0%) 的利息，即 $0.729 375M. 相当于 A 等级的 IO 类型的名义本金为 $9.725M，收益率为 7.5%. 每个等级的名义本金之和等于 IO 的名义本金 $13.75M，如表 3.18 所示.

表 3.18

份 额	面 值	PT 率	(0.075−PT 率) 面值	名义本金
A	$48.625M	6.0%	$729 375	$9.725M
B	$9M	6.5%	$90 000	$1.2M
Z	$42.375M	7.0%	$211 875	$2.825M
			合计	$13.75M

假设我们有价值为 30 000 000 美元的抵押贷款组合，分 A、B、C 三个份额，每个份额价值为 10 000 000 美元. 假设第一份额在 60 个月内全部偿付，第二份额在 90 个月内全部偿付，第三份额按规则摊销至到期日. 每一份额的提前还款速率分别为 100、165 和 200. 假设发行后第一次付款的延迟日期为 30 天，WAC (Matlab 中的 GrossRate) 为 8.125%，PT 率 (Matlab 中的 CouponRate) 为 7.5%，发行日是 2004 年 3 月 1 日，交割日 (结算日) 是 2004 年 3 月 1 日，到期日是 2034 年 3 月 1 日. 用以下 Matlab 命令计算交割日和到期日之间的现金流，交割日的相应时间因子，以及每一份额的抵押贷款组合因子 (未偿还贷款本金的因子)：

```
% mbsfamounts
% [output] CFlowAmounts: vector of cash flows starting from Settle
% through end of the last month (Maturity)
%                CflowDates: indicates when cash flows occur, including
% at Settle.  A negative number at Settle indicates
% accrued interest is due.
%                TFactors: vector of times in months from Settle,
% corresponding to each cash flow.
%                Factors: vector of mortgage factors (the fraction of
% the balance still outstanding at the end of each month).
Settle = [datenum('1-Mar-2004');
              datenum('1-Mar-2004');
              datenum('1-Mar-2004')];
Maturity = [datenum('1-Mar-2034')];
IssueDate = datenum('1-Mar-2004');
GrossRate = 0.08125;
CouponRate = 0.075;
Delay = 30;
PSASpeed = [100; 165; 200];
[CPR, SMM] = psaspeed2rate(PSASpeed);
PrepayMatrix = ones(360,3);
PrepayMatrix(1:60,1) = SMM(1:60,1);
PrepayMatrix(1:90,2) = SMM(1:90,2);
PrepayMatrix(:,3)=SMM(:,3)
 [CFlowAmounts, TFactors, Factors] = mbscfamounts(Settle, Maturity,
    IssueDate, ...
  GrossRate, CouponRate, Delay, [], PrepayMatrix)
```

不同顺序份额的现金流如表 3.19 所示.

表 3.19

月	A 份额的现金流（CF）	B 份额的现金流（CF）	C 份额的现金流（CF）
1	0	0	0
2	70 708.00	71 794.00	72 379.00
3	72 368.00	74 534.00	75 702.00
4	74 015.00	77 256.00	79 004.00
5	75 649.00	79 957.00	82 283.00
59	95 117.00	105 150.00	108 680.00
60	94 592.00	104 190.00	107 470.00
61	94 070.00	103 240.00	106 270.00
62	7 580 500.00	102 290.00	105 080.00
63	0.00	101 350.00	103 910.00
64	0.00	100 420.00	102 750.00
88	0.00	80 383.00	78 340.00
89	0.00	79 637.00	77 455.00
90	0.00	78 897.00	76 579.00
91	0.00	78 163.00	75 713.00
92	0.00	4 811 500.00	74 857.00
93	0.00	0.00	74 009.00
358	0.00	0.00	2 017.40
359	0.00	0.00	1 976.90
360	0.00	0.00	1 936.80
361	0.00	0.00	1 897.30

我们可以用以下代码计算每个抵押贷款组合的价格和累计利息：

```
[Price, AccrInt] = mbsprice(Yield, Settle, Maturity, IssueDate,
GrossRate, CouponRate, Delay, PrepaySpeed, PrepayMatrix)
```

价格和累计利息如表 3.20 所示. 因为交割日和发行日一致，没有一个份额有累计利息.

我们可以利用 mbsdurp（价格给定下的久期）、mbsdury（收益给定下的久期）、mbsconvp（价格给定下的凸性）和 mbsconvy（收益给定下的凸性）来计算抵押贷款组合的有效久期和凸性. 例如，接着上一个例子，我们可以用以下命令计算在 PSA 速率分别为 100、150、200 的假设下组合在 2004 年 3 月 10 日的年久期、修正久期和凸性：

```
% compute regular duration and modified duration
[YearDuration, ModDuration] = mbsdurp(Price, Settle, Maturity,
IssueDate, GrossRate,
   CouponRate, Delay, PrepaySpeed)

% compute convexity
Convexity = mbsconvp(Price, Settle, Maturity, IssueDate, GrossRate,
CouponRate, Delay,
    PrepaySpeed)
```

对于 100、150 和 200 的 PSA，上面 Matlab 函数获得的久期、修正久期和凸性结果如表 3.21 所示. 图 3.6 显示一个按次序偿还的 CMO 的模拟现金流.

表 3.20

	价 格	累计利息
份额 A	101.147 7	0
份额 B	100.852 0	0
份额 C	100.731 1	0

表 3.21

PSA	年久期	修正久期	凸 性
100	7.066 9	6.814 8	82.323 0
150	6.104 8	5.888 1	62.247 6
200	5.371 2	5.181 4	48.336 8

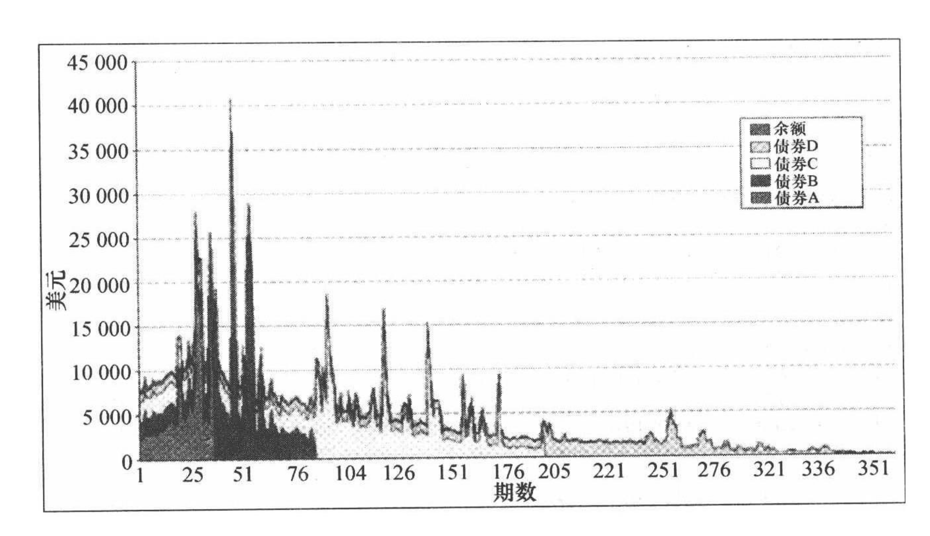


图 3.6 CMO 的模拟现金流

资料来源：Bandic(2002)，28.

3.8 CMO 在 C++ 中的应用

CMO.h

```
#ifndef _CMO__
#define _CMO__

#include <vector>
#include <algorithm>
#include <numeric>
#include "Constants.h"
#include "MBS.h"

using namespace std;

class Tranche
{
  public:
        Tranche() {}
        Tranche(char clas, double balance, double coupon)
          : initBalance_(balance), balance_(balance), coupon_(coupon),
              clas_(clas) {}
        virtual ~Tranche() {}
        double initBalance_;
        double balance_;
        double coupon_;
        vector<double> cashFlows_;
        vector<double> sumCF_;
        vector<double> inter_;
        vector<double> principal_;
        vector<double> discount_;
        vector<double> T_;
        double price_;
        double interest_;
        double princip_;
        double averageLife_;
        char clas_;
};

class CMO
{
  public:
        CMO(MBS m, vector<Tranche> tr) : mbs(m), tranche(tr)
        {
          for (int i = 0; i < tranche.size(); i++)
                collateral_.push_back(tranche[i].balance_);
        }
        virtual ~CMO() { }
        void calcTrancheCF();
        inline double calcCPR(double SMM) { return 100*(1-pow((1-
           (SMM/100)),12)); }
        inline double calcSMM(double scheduleBal, double actualBal) {
          return 100*((double)(scheduleBal - actualBal)/scheduleBal);
        }
        inline double calcPSA(double age, double CPR) {

          return 100*((double)(CPR/(0.2*min(age,30))));
    }
    inline double calcRefinance(double r) {
```

```
    double WAC = mbs.getWAC();
    double a = (double) 0.5/2;
    double b = (double) 100*((0.5 - a)/(PI/2));
    double d = (double) 0.06/b;
    double c = (double) -d*0.02;
    return (double) (a + b*(atan(c + d*(WAC - r))));
  }
  inline double calcBurnout(int t, Tranche tr, double balance) {

    return (double) (0.3 + 0.7*((double)balance/1000000));
  }
  inline double calcMP(int t, Tranche tr, double balance)  {

    double WAC = mbs.getWAC();
    double WAM = mbs.getWAM();

    return balance*(((double)WAC/12)/(1-pow((1+(double)WAC/12),
       -WAM+t)));
  }
  inline double calcIP(int t, Tranche tr, double r,
     double balance) {

    double WAC = mbs.getWAC();

    return (balance)*((double)(tr.coupon_/12));
  }
  inline double calcPP(int t, Tranche tr, double r,
     double balance) {

    double SMM = calcSMM1(t,tr,r,balance);
    double SP = calcSP(t,tr,r,balance);

    return SMM*(balance - SP);

  }
  inline double calcMM(int t) {

    double MM[12] = { 0.94, 0.76, 0.74, 0.95, 0.98, 0.92, 0.98,
                       1.10, 1.18, 1.22, 1.23, 0.98};
    int rem =  t % 12;

    if (t == 1)
          rem = 1;

    return MM[rem-1];

  }
  inline double calcCPR1(int t, Tranche tr, double r,
     double balance) {

    double RI = calcRefinance(r);
    double age = calcAge(t);
    double MM = calcMM(t);
    double BM = calcBurnout(t,tr,balance);

    return RI*age*MM*BM;

  }
  inline double calcAge(int t) {
    return min((double)t/30,1);
```

```
        }
        inline double calcSMM1(int t,Tranche tr, double r,
           double balance) {

          double CPR = calcCPR1(t,tr,r,balance);

          return (1 - pow((1 - CPR),(double)1/12));
        }
        inline double calcSP(int t, Tranche tr, double r,
           double balance) {

          double MP = calcMP(t,tr,balance);
          double IP = calcIP(t,tr,r,balance);

          return MP - IP;
        }
        void calcCashFlows(double initRate, double financeRate, int N,
           int M);
  private:
        MBS mbs;
        vector<Tranche> tranche;
        vector<double> collateral_;
};

#endif
```

方法定义如下：

CMO.cpp

```
#include "CMO.h"
#include "Utility.h"

void CMO::calcCashFlows(double initRate, double financeRate, int N, int M)
{
   Utility util;
   int i, t = 0;
   double r = 0.0715;
   const double kappa = 0.29368;
   const double vol = 0.11;
   const double theta = 0.08;
   double deviate = 0;
   long seed = 0;
   long* idum = 0;
   double balance = 0;
   double sum = 0;
   double S[4] = {0};
   double sum1 = 0;
   double sum2 = 0;
   double sum3 = 0;
   double sumA = 0;
   double sumB = 0;
   double sumC = 0;
   double sumD = 0;
   double CPR = 0;
   double interest = 0.0;
   double mbsPrice = 0;
   double stdErr = 0;
   double stdDev = 0;
   double totalsum = 0;
```

```
double totalsumA = 0;
double totalsumB = 0;
double totalsumC = 0;
double totalsumD = 0;
double totalsum2 = 0;
double schedulePrincipal = 0;
double prepaidPrincipal = 0;
double discount = 0;
double principal = 0;
double pay = 0.0;
double r1 = 0.0;
double rr = 0.0;
int cnt = 0;
double trancheBal = 0.0;
double T = mbs.getMaturity();
double WAM = mbs.getWAM();
double OAS = mbs.getOAS();
double dt = (double) T/N;
double interest1 = 0;
vector<double> disc(0.0);
vector<double> time1;
TNT::Array1D<double> CF(SIZE_X);   // cash_flow
vector<double> p;

srand(unsigned(time(0)));
seed = (long) rand() % 100;
idum = &seed;

for (t = 1; t <= N; t++)
  time1.push_back((double)(t-1)/12);

for (i = 0; i < M; i++)
{
  r = initRate;
  sum = 0;
  sumA = 0;
  sumB = 0;
  sumC = 0;
  sumD = 0;
  schedulePrincipal = 0;
  prepaidPrincipal = 0;
  balance = 1000000;
  cnt = 1;
  disc.clear();
  disc.empty();
  disc.push_back(r);
  p.clear();
  p.push_back(0);

  for (int j = 0; j < tranche.size(); j++)
  {
    tranche[j].balance_ = collateral_[j];
    tranche[j].inter_.clear();
    tranche[j].principal_.clear();
    trancheBal = calcPP(0,tranche[j],r,tranche[j].balance_) +
       calcMP(0,tranche[j],tranche[j].balance_);
    tranche[j].principal_.push_back(trancheBal);
    tranche[j].interest_ = calcIP(0,tranche[j],r,tranche[j].balance_);
    tranche[j].inter_.push_back(tranche[j].interest_);
    S[j] = 0;
```

```
    }

    for (t = 1; t <= N; t++)
    {
      balance = balance - (schedulePrincipal + prepaidPrincipal);
      deviate = util.gasdev(idum);
      r = r + kappa*(theta - r)*dt + vol*r*sqrt(dt)*deviate;
      disc.push_back(r);
      interest = calcIP(t,tranche[cnt-1],r,balance);
      schedulePrincipal = calcMP(t,tranche[cnt-1],balance);
      prepaidPrincipal = calcPP(t,tranche[cnt-1],r,balance);
      tranche[cnt-1].balance_ = tranche[cnt-1].balance_ -
         schedulePrincipal - prepaidPrincipal;
      principal = schedulePrincipal + prepaidPrincipal;
      tranche[cnt-1].principal_.push_back(principal);
      tranche[cnt-1].princip_ = principal;
      p.push_back(principal);

      if (tranche[cnt-1].balance_ > 0)
        interest1 = calcIP(t,tranche[cnt-1],r,tranche[cnt-1].balance_);
      else
        interest1 = 0;

      tranche[cnt-1].inter_.push_back(interest1);
      tranche[cnt-1].interest_ = interest1;

      for (int k = 1; k <= tranche.size(); k++)
      {
        if (k != cnt)
        {
           interest1 = calcIP(t,tranche[k-1],r,tranche[k-1].balance_);

           if (tranche[k-1].balance_ != 0 )
           {
             tranche[k-1].inter_.push_back(interest1);
             tranche[k-1].interest_ = interest1;
           }
           else
           {
             tranche[k-1].inter_.push_back(0.0);
             tranche[k-1].interest_ = 0.0;
           }

           tranche[k-1].principal_.push_back(0.0);
           tranche[k-1].princip_ = 0.0;
        }

        rr = mbs.computeZeroRates(t-1,disc);
        S[k-1] = (tranche[k-1].interest_ + tranche[k-1].princip_)/
           (pow(1+rr+OAS,(double)(t-1)/12));

        if (k == 1)
          sumA = sumA + S[k-1];
        else if (k == 2)
          sumB = sumB + S[k-1];
        else if (k == 3)
          sumC = sumC + S[k-1];
        else
          sumD = sumD + S[k-1];
      }
```

```
        if (tranche[cnt-1].balance_ > 0)
        {
          if (balance >= schedulePrincipal)
          {
            if (t != N)
               CF[t-1] = schedulePrincipal + interest + prepaidPrincipal;
            else
               CF[t-1] = interest + balance;

            rr = mbs.computeZeroRates(t-1,disc);
            sum = sum + CF[t-1]/(pow(1+rr+OAS,(double)(t-1)/12));
          }
          else
            goto x;
        }
        else
        {
          tranche[cnt-1].balance_ = 0;
          cnt++;
        }
      }
      x:
        totalsum = totalsum + sum;
        totalsumA = totalsumA + sumA;
        totalsumB = totalsumB + sumB;
        totalsumC = totalsumC + sumC;
        totalsumD = totalsumD + sumD;
        totalsum2 = totalsum2 + sum*sum;
  }

  calcTrancheCF();
  for (int j = 0; j < tranche.size(); j++)
  {
        sum1 = 0;
        sum2 = 0;
        for (i = 0; i < tranche[j].principal_.size(); i++)
        {
          sum1 = sum1 + (time1[i])*(tranche[j].principal_[i]);
          sum2 = sum2 + tranche[j].principal_[i];
        }
        tranche[j].averageLife_ = sum1/sum2;
  }
  sum1 = 0;
  sum = accumulate(p.begin(),p.end(),0);
  for (j = 0; j < p.size(); j++)
        sum1 = sum1 + time1[j]*p[j];

  std::cout << endl;
  std::cout << "collateral price = " << totalsum/M <<  " " << "Ave.Life = "
     << sum1/sum << endl;
  std::cout << "Tranche A price =  " << totalsumA/M << " " << "Ave.Life = "
     << tranche[0].averageLife_ << endl;
  std::cout << "Tranche B price =  " << totalsumB/M << " " << "Ave.Life = "
     << tranche[1].averageLife_ << endl;
  std::cout << "Tranche C price =  " << totalsumC/M << " " << "Ave.Life = "
     << tranche[2].averageLife_ << endl;
  std::cout << "Tranche Z price =  " << totalsumD/M << " " << "Ave.Life = "
     << tranche[3].averageLife_ << endl;

  T = mbs.getMaturity();
```

```
  stdDev = sqrt(totalsum2 - (double)(totalsum*totalsum)/M)*
     (exp(-2*initRate*T)/(M-1));
  stdErr = (double) stdDev/sqrt(M);
}

void CMO::calcTrancheCF()
{
  vector<Tranche>::iterator iter;
  vector<double>::iterator iter1;
  vector<double>::iterator iter2;
  int cnt = 1;

  for (iter = tranche.begin(); iter != tranche.end(); iter++)
  {
     iter2 = iter->inter_.begin();
        cnt = 1;
        for (iter1 = iter->principal_.begin(); iter1 !=
           iter->principal_.end(); iter1++)
        {
          std::cout << "Mo." << cnt << "  Class: " << iter->clas_
            << " " << "Principal= " << *iter1
            << " " << "Coupon= " << *iter2 << endl;
          iter2++;
          cnt++;
        }
  }
}
```

主要方法如下：

Main.cpp

```
void main()
{

    std::cout.precision(7);
    double principal = 1000000;        // underlying principal notional)
                                       // of MBS
    double coupon = 0.08;              // coupon rate
    double WAC = 0.08;                 // weighted average coupon rate
    double WAM = 10;                   // weighted average maturity
    double OAS = 0.02;                 // option adjusted spread
    double initSpotRate = 0.06;        // spot rate
    double initRefinanceRate = 0.08;   // refinance rate
    int N = 10;                        // number of time steps in tree
    long int M = 100000;               // number of simulation paths

    MBS mbs(principal,coupon,WAC,WAM,OAS);

    vector<Tranche> tranche;
    Tranche trA('A',500000,0.06);
    tranche.push_back(trA);
    Tranche trB('B',300000,0.065);
    tranche.push_back(trB);
    Tranche trC('C',200000,0.07);
    tranche.push_back(trC);
    Tranche trZ('Z',100000, 0.075);
    tranche.push_back(trZ);

    std::cout << endl;
```

```
    std::cout << "Pricing CMO Tranches..." << endl << endl;
    CMO cmo(mbs,tranche);
    cmo.calcCashFlows(initSpotRate,initRefinanceRate,N,M);

}
```

结果如下：

```
Pricing CMO Tranches...

Mo.1    Class: A Principal= 51851.6   Coupon= 2500
Mo.2    Class: A Principal= 115430    Coupon= 1922.85
Mo.3    Class: A Principal= 114668.5 Coupon= 1349.507
Mo.4    Class: A Principal= 113798.2 Coupon= 780.5165
Mo.5    Class: A Principal= 113024    Coupon= 215.3967
Mo.6    Class: A Principal= 111815.5 Coupon= 0
Mo.7    Class: A Principal= 0         Coupon= 0
Mo.8    Class: A Principal= 0         Coupon= 0
Mo.9    Class: A Principal= 0         Coupon= 0
Mo.10   Class: A Principal= 0         Coupon= 0
Mo.1    Class: B Principal= 31110.96 Coupon= 1625
Mo.2    Class: B Principal= 0         Coupon= 1625
Mo.3    Class: B Principal= 0         Coupon= 1625
Mo.4    Class: B Principal= 0         Coupon= 1625
Mo.5    Class: B Principal= 0         Coupon= 1625
Mo.6    Class: B Principal= 0         Coupon= 1625
Mo.7    Class: B Principal= 110373.5 Coupon= 1027.143
Mo.8    Class: B Principal= 108932.9 Coupon= 437.0903
Mo.9    Class: B Principal= 107334.2 Coupon= 0
Mo.10   Class: B Principal= 0         Coupon= 0
Mo.1    Class: C Principal= 20740.64 Coupon= 1166.667
Mo.2    Class: C Principal= 0         Coupon= 1166.667
Mo.3    Class: C Principal= 0         Coupon= 1166.667
Mo.4    Class: C Principal= 0         Coupon= 1166.667
Mo.5    Class: C Principal= 0         Coupon= 1166.667
Mo.6    Class: C Principal= 0         Coupon= 1166.667
Mo.7    Class: C Principal= 0         Coupon= 1166.667
Mo.8    Class: C Principal= 0         Coupon= 1166.667
Mo.9    Class: C Principal= 0         Coupon= 1166.667
Mo.10   Class: C Principal= 105320.6 Coupon= 552.2968
Mo.1    Class: Z Principal= 10370.32 Coupon= 625
Mo.2    Class: Z Principal= 0         Coupon= 625
Mo.3    Class: Z Principal= 0         Coupon= 625
Mo.4    Class: Z Principal= 0         Coupon= 625
Mo.5    Class: Z Principal= 0         Coupon= 625
Mo.6    Class: Z Principal= 0         Coupon= 625
Mo.7    Class: Z Principal= 0         Coupon= 625
Mo.8    Class: Z Principal= 0         Coupon= 625
Mo.9    Class: Z Principal= 0         Coupon= 625
Mo.10   Class: Z Principal= 0         Coupon= 625

collateral price = 682754.6 Ave.Life = 0.4104376
Tranche A price =  564751.1 Ave.Life = 0.2279203
Tranche B price =  321741.6 Ave.Life = 0.5318972
Tranche C price =  108757.8 Ave.Life = 0.6266037
Tranche Z price =  5460.491 Ave.Life = 0
```

3.9 计划摊销份额（PAC）

计划摊销份额（Planned Amortization Class，PAC）（通常还叫做计划赎回义务）是指一些无（或至少是最小的）提前偿还风险的份额. PAC 是通过对抵押品实施低和高的 PSA 速率而建立起来的. 因而 PAC 债券每月可以得到一个最低的现金流保证，同时还有一个支持性债券，获得余下现金流. 这些支持性债券，有时被称为伴随性债券，吸收本金偿还，如果 PSA 超过 PAC 的范围则会比计划的偿还速度更快. 如果 PSA 低于这个范围，则支持性债券的寿命和摊还期更长. 在任何情况下，由于支持性债券提供的稳定性，PAC 的波动性比按次序偿还结构的波动性要小.

只要 PSA 速率在使用的低 PSA 阈值和高 PSA 阈值之间，则 PAC 对提前偿还风险的敏感性要比标准债券低. 假设我们对一个 WAC=8.125%，WAM=357 个月，而 PT 率=7.5%的 $100M 的抵押组合的抵押品实施 90 和 300 的 PSA 模型. 这个会得出整个 357 个月的两个不同的月本金. 我们还需要将一个季度因子计入考虑，这个因子可以计量基于季度的提前还款，我们假设其为 3. 表 3.22 显示具有以上性质的 PAC 的现金流.

表 3.22

PAC 期数	PAC 低 PSA Pr	PAC 高 PSA Pr	PAC 最小本金	PAC 利率 0.08	抵押余额 100.000 000 00	抵押利息	抵押本金
1	127 042.38	268 982.80	127 042.38	381 088.87	100 000 000.00	625 000.00	177 480.87
2	142 460.86	319 853.03	142 460.86	380 294.85	99 822 519.13	623 890.74	205 473.91
3	157 844.28	370 548.61	157 844.28	379 404.47	99 617 045.22	622 606.53	233 389.97
4	173 185.47	420 991.54	173 185.47	378 417.95	99 383 655.25	621 147.85	261 205.39
5	188 477.28	471 103.66	188 477.28	377 335.54	99 122 449.86	619 515.31	288 896.53
6	203 712.56	520 806.83	203 712.56	376 157.55	98 833 553.33	617 709.71	316 439.78
7	218 884.17	570 023.17	218 884.17	374 884.35	98 517 113.54	615 731.96	343 811.60
102	363 181.39	371 031.65	363 181.39	129 910.54	41 682 140.85	260 513.38	421 018.70
103	361 690.25	364 654.51	361 690.25	127 640.65	41 261 122.15	257 882.01	417 259.77
104	360 206.39	358 384.81	358 384.81	125 380.09	40 843 862.38	255 274.14	413 533.58
105	358 729.77	352 220.75	352 220.75	123 140.18	40 430 328.80	252 689.56	409 839.84
106	357 260.36	346 160.58	346 160.58	120 938.80	40 020 488.96	250 128.06	406 178.29
107	355 798.12	340 202.57	340 202.57	118 775.30	39 614 310.67	247 589.44	402 548.63
356	150 750.84	2 573.92	2 573.92	31.76	74 797.79	467.49	37 597.30
357	150 372.35	2 507.45	2 507.45	15.67	37 200.49	232.50	37 200.49

资料来源：Johnson S(2004).

为演示计算过程，第一个月的 PAC 余额为 100 000 000 美元. 第一个月的利息为：

$$I_1 = (0.075/12) * (100\,000\,000) = 625\,000 \text{ 美元}$$

PAC 的本金偿还金额是：

$$p = \frac{(0.081\,25/12)(100\,000\,000)}{1-(1/(1+0.081\,25/12))^{357}} = 743\,967.06 \text{ 美元}$$

PAC 的计划本金偿还金额是：

$$743\,967.06 - (0.085/12)(100\,000\,000) = 66\,883.73 \text{ 美元}$$

假设 PAC 的 PSA 低速率为 90. 我们计算 PAC 的季度调整因子：

$$\min\left(\frac{\text{月份}+\text{季度因子}}{\text{到固定 CPR 时的月数}}, 1\right)$$

因此，第一个月的季度调整因子为：

$$\min\left(\frac{1+3}{30},1\right)=0.133\,33$$

然后对 PAC 的 CPR 使用 PAC 的季度调整因子：

$$\left(\frac{\text{PSA}}{100}\right)(\text{固定 CPR})(\text{季度调整因子})=\left(\frac{90}{100}\right)(0.06)(0.133\,3)=0.007\,2$$

SMM 为：

$$\text{SMM}=(1-(1-\text{PAC CPR}))^{1/357}=(1-(1-0.007\,2))^{1/357}=0.000\,601\,99$$

PAC 提前偿还本金金额为：

$$\text{SMM}(\text{余额}-\text{计划本金})\cdot 0.000\,601\,99(100\,000\,000-66\,883.73)=\$60\,158.65$$

PAC 现金流为：

$$\begin{aligned}\text{PAC 利息}+\text{PAC 计划本金}+\text{PAC 提前偿还本金}&=625\,000+66\,883.73+60\,148.65\\&=\$752\,042.38\end{aligned}$$

我们现在可以计算低 PSA 速率下 PAC 的提前偿还总额：

$$\text{PAC 计划本金}+\text{PAC 提前偿还本金}=66\,883.72+60\,158.65=\$127\,042.38$$

计算过程对高水平 PSA=300 同样适用. 主要的不同是 PAC CPR：

$$\left(\frac{\text{PSA}}{100}\right)(\text{固定 CPR})(\text{季度调整因子})=\left(\frac{300}{100}\right)(0.06)(0.133\,3)=0.024$$

这个 PAC 债券的平均寿命为 7.26 年. 另外，在 PSA 速率为 90 到 300 之间时，PAC 债券的平均寿命为 7.26 年，这意味着没有提前还款风险. 表 3.23 显示在很多不同 PSA 假设下的 PAC 平均寿命和支持性债券.

表 3.23

PSA 速率	PAC	支持性债券	担 保	PSA 速率	PAC	支持性债券	担 保
0	10.36	23.84	20.36	200	7.26	8.36	7.69
50	8.04	21.69	15.36	250	7.26	5.35	6.52
90	7.26	20.06	12.26	300	7.26	3.11	5.64
100	7.26	18.56	11.67	350	6.61	2.91	4.98
150	7.26	12.56	9.33	400	6.06	2.74	4.45

PAC 债券可以分解为其他的 PAC 份额. 最常见的是按次序还款的 PAC. 例如，我们可以使用之前的抵押品来组合 6 个按次序还款 PAC. PSA 在区间 90～300 之间时，PAC 级别的平均寿命稳定；90～300 称为领（collar）. 一些 PAC 会游离到区间之外，这被称为有效领（effective collar）. 拥有的级别越多，窗口越窄，就使得 PAC 像一个子弹式债券. 这样的 PAC 应该被卖给负债管理基金来使其负债与特定的负债或久期相符合，这就是现金流匹配. 在 20 世纪 80 年代，我们最多可以找到有 70 个级别的 CMO（尤其是 PAC）. 在 20 世纪 90 年代早期，平均级别数量是 24 个. 像 PAC 一样，支持性债券也可以被分为不同的级别：按次序还款、流动性和累计债券等.

在一定 PSA 区间范围内目标摊还份额（Targeted Amortization Class，TAC）也提供提前还款保护，但是这个 PSA 不低于用来为 CMO 定价的 PSA. 提前还款速度放慢会导致平均寿命

延长，因此，TAC 的收益比 PAC 更高.

图 3.7 展示了支付给一个假设 GNMA PAC 100/300 的现金流.

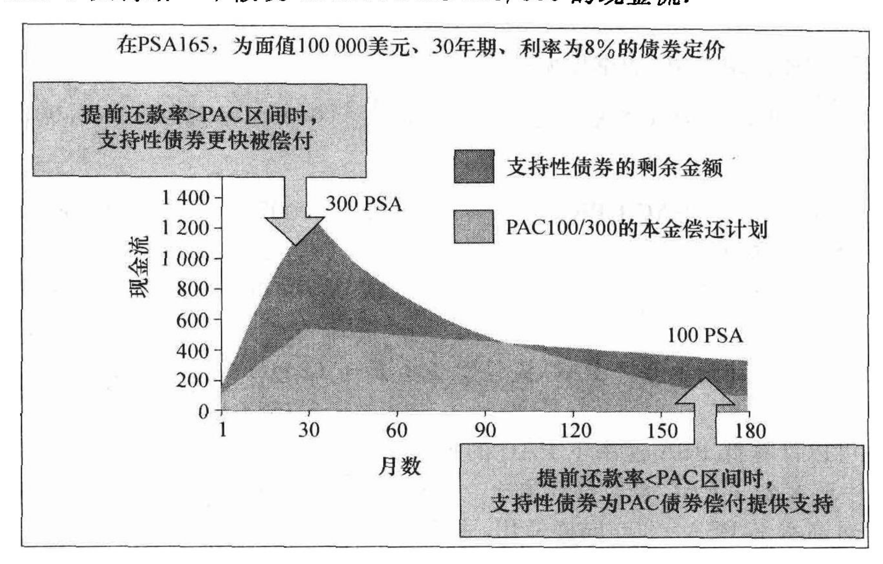


图 3.7 GNMA PAC 100/300 的现金流

3.10 纯本金债券与纯利息债券

剥离式 MBS 是由 FNMA 在 1986 年提出的. 任何 MBS 都可以通过将担保资产现金流分为纯本金（PO）或纯利息（IO）证券来进行剥离（stripped）并分别销售. 纯利息债券只收到抵押贷款的利息. 纯本金债券只收到本金的偿还. 纯本金债券的回报依赖于提前偿还的速率. 提前偿还越快，收益越高. 例如，纯本金投资者投资 \$75M 购买了一个抵押贷款组合，其本金为 \$100M，如果这 \$100M 提前偿还（比如提前几年偿还），则投资者将获得更高收益. 纯本金债券有一个逆价格-利率敏感性的关系：如果利率降低，则提前还款增加，从而纯本金的收益增加，价格（价值）增加. 类似地，如果利率升高，则提前还款减少，从而纯本金的（收益）回报减少，价格（价值）下降.

因为纯利息投资者仅获得本金的利息，他们希望提前还款速度缓慢. 例如，纯利息投资者持有一个本金为 \$400M、利率为 7.5% 的组合的纯利息债券，如果本金立即偿付，可以得到 \$30M（= \$400 * 0.075）. 相反，如果本金在 4 年内以相等的增量付清，则回报为 \$75M（见表 3.24）.

表 3.24

1 年	(\$400M)(0.75) = \$30.0M
2 年	(\$300M)(0.75) = \$22.5M
3 年	(\$200M)(0.75) = \$15.0M
4 年	(\$100M) (0.75) = \$7.5M
合计	\$75M

因为利率降低会使提前偿还速率上升，从而降低了纯利息债券的收益，进而导致其价值降低. 纯利息债券的价格是否随利率降低而降低取决于这个效应是否超过贴现率降低对债券价值上升的效应. 换句话说，当利率降低时，提前还款金额增加，纯利息债券收益下降，并且纯利息债券的价值必须被贴现率降低带来的价值增

加平衡. 这两个效应可能会互相抵消，以至于纯利息债券的价值与收益之间可能存在直接关系.

像 CMOS 一样，剥离式 MBS 是一个衍生产品. 每个剥离都极具波动性，并且如上所述，取决于提前还款速率. 当本金是以票面值的折扣价购买，而最终还款额等于票面值，还款比预期要快时，纯本金债券在提前还款金额很高的情况下表现很好，这使它们成为一个具有很大的正久期的牛市投资. 纯利息债券在一个提前还款速率很低的情况下表现更好，因为本金产生利息的时间更长，而且利息的支付是持续的，这使其成为一个久期为负的投资，即它们的价格随着利率的上升而上升，反之亦然. 纯利息债券通常用来对冲 MBS 或 CMO 组合的利率风险. 利率上升带来的损失可以部分或全部被纯利息债券的相应升值所抵消，这取决于对冲结构.

考虑一个剥离式 MBS，其抵押组合价值为 $100M，WAM=357 个月，WAC=0.081 25，PT 率=0.075，PSA=165. 现金流如表 3.25 所示.

表 3.25

期数	余额 100 000 000	利息	计划本金	提前偿还本金	本金	剥离式 MBS 纯本金债券	纯利息债券
1	100 000 000	625 000	65 216. 471 56	110 598. 991 1	175 815. 462 7	175 815. 462 7	625 000
2	99 824 184. 54	623 901. 153	65 592. 162 76	138 216. 466 5	203 808. 629 3	203 808. 629 3	623 901. 2
3	99 620 375. 91	622 627. 349	65 951. 605 53	165 774. 637 1	231 726. 242 6	231 726. 242 6	622 627. 3
4	99 388 649. 67	621 179. 06	66 294. 447 44	193 250. 204	259 544. 651 5	259 544. 651 5	621 179. 1
5	99 129 105. 01	619 556. 906	66 620. 346 73	220 619. 862	287 240. 208 8	287 240. 208 8	619 556. 9
6	98 841 864. 81	617 761. 655	66 928. 972 88	247 860. 331 9	314 789. 304 7	314 789. 304 7	617 761. 7
7	98 527 075. 5	615 794. 222	67 220. 007 09	274 948. 392 9	342 168. 4	342 168. 4	615 794. 2
104	40 930 020. 29	255 812. 627	59 884. 949 69	353 521. 540 6	413 406. 490 3	413 406. 490 3	255 812. 6
105	40 516 613. 8	253 228. 836	59 775. 100 19	349 946. 576 6	409 721. 676 8	409 721. 676 8	253 228. 8
106	40 106 892. 12	250 668. 076	59 665. 452 19	346 403. 483 9	406 068. 936 1	406 068. 936 1	250 668. 1
107	39 700 823. 19	248 130. 145	59 556. 005 32	342 891. 985 3	402 447. 990 6	402 447. 990 6	248 130. 1
108	39 298 375. 2	245 614. 845	59 446. 759 22	339 411. 805 7	398 858. 564 9	398 858. 564 9	245 614. 8
109	38 899 516. 63	243 121. 979	59 337. 713 51	335 962. 672 4	395 300. 385 9	395 300. 385 9	243 122
110	38 504 216. 25	240 651. 352	59 228. 867 83	332 544. 315 2	391 773. 183	391 773. 183	240 651. 4
111	38 112 443. 06	238 202. 769	59 120. 221 81	329 156. 466 1	388 276. 687 9	388 276. 687 9	238 202. 8
112	37 724 166. 38	235 776. 04	59 011. 775 08	325 798. 859 5	384 810. 634 6	384 810. 634 6	235 776
113	37 339 355. 74	233 370. 973	58 903. 527 28	322 471. 232 2	381 374. 759 4	381 374. 759 4	233 371
356	75 665. 721 72	472. 910 761	377 03. 255 92	328. 370 564 1	38 031. 626 48	38 031. 626 48	472. 910 8
357	37 634. 095 24	235. 213 095	37 634. 095 24	−3. 020 93E-12	37 634. 095 24	37 634. 095 24	235. 213 1

3.11 利率风险

MBS 利率风险与其他固定收益证券的利率风险相似：利率下降，价格上升，反之亦然. 然而，基于证券息票率和现行抵押贷款率之间的关系，MBS 内含的提前还款期权会影响价格变动程度. 当息票率等于或高于现行抵押贷款率时，业主更可能执行他们的提前还款期权. 随着提前还款可能性的增加，由于提前还款风险上升，MBS 价格上升得没有其他相似但不含期权的证券多. 这被称为*负凸性* (negative convexity).

当息票率低于现行抵押贷款率或处于“虚值”时，业主不可能执行提前还款期权. 尽管这

个提前还款期权不太可能被执行，但债券价格仍然显示为负凸性，原因在于投资者在比现行市场利率低的利率水平上维持本金投资更长时间. 这就是**延期风险**（extension risk）. 图 3.8 显示了一般固定收益证券价格和利率之间的反向关系. 图 3.9 表示了 MBS 的负凸性.

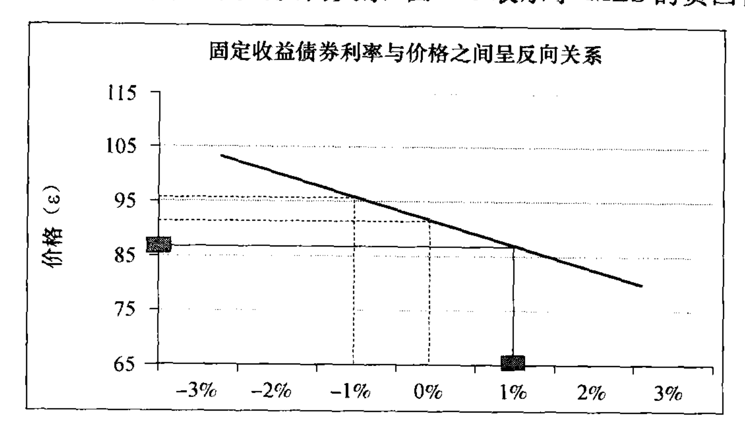


图 3.8 一般固定收益债券价格和利率之间的反向关系

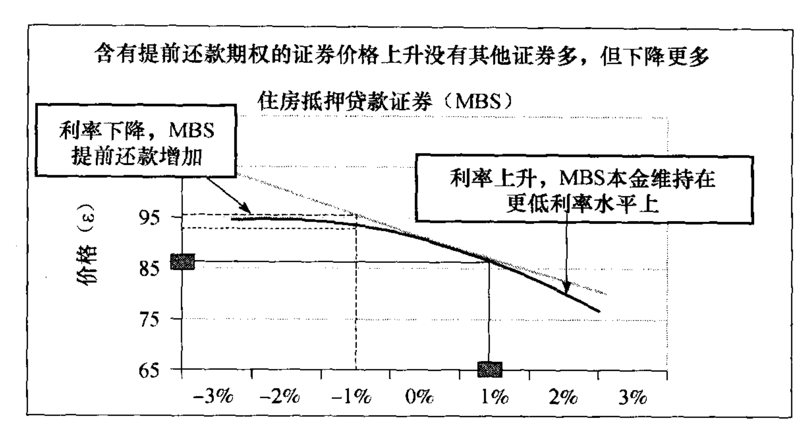


图 3.9 MBS 的负凸性

3.12 MBS 的动态对冲

机构重仓持有住房抵押贷款证券（MBS）是有很多原因的. 其中对冲 MBS 的利率风险是对这一持仓是否能够反映因主营业务的相关价值或持有库存的交易的一个很重要的考虑. MBS 的对冲是很复杂的，其对现金流的预期依赖于整个组合的提前还款行为. 尤其是在利率下降时，抵押贷款者因为再融资动机而提前还款的可能性更高. 因此，固定利率投资者在其相应固定利率债券上卖出了一个潜在的看涨期权[26]. 尽管其他因素也会影响提前还款（比如季节性和衰减），但是利率还是 MBS 定价的最重要因素. 因为这个最重要性质，美国中期国债（T-note），或者更确切地，中期国债期货经常用来对冲 MBS. 这里有两个原因：（1）中期国债期货很容易变现；（2）这些金融工具的价格是由其利率的期限结构来决定的，因此直接与 MBS 价值相关[27]. 我们在以下的讨论中按照 Boudoukh、Richardson、Stanton 和 Whitelaw（1995）来进行.

用中期国债期货来对冲 MBS 有两种常用方法. 第一是纯经验的，涉及 MBS 的收益关于中期国债期货收益的回归. 从得到的关系估计的回归系数可以在用中期国债风险对冲 MBS 的利率风险时被利用. 这种方法的优点是对于相关的利率或提前还款的演变模型没有很强的假设存在[28]. 缺点是这种方法是静态的. 它不会对利率变化和抵押贷款提前还款金额的对冲比率进行外部调整，这可能导致错误对冲一个很大投资组合的潜在不利[29]. 因此，在回归中使用的观测值仅代表样本期内 MBS 和中期国债期货之间的关系均值，可能代表也可能不代表当前时期.

作为另一个选择，第二种方法是基于模型的. 其包含了对利率加工的详述和一个提前还款模型. 假设条件帮助建立 MBS 价格关于利率和其他可能要素的函数[30]. 这种方法代表了一个用来决定 MBS 定价和中期国债期货定价之间的共同活动的动态算法. 这些共同活动完全按照相关经济变量的当前值和特别参数值分辨开. 基本思想是估计 MBS 收益和中期国债收益之间的条件对冲比率. 这对 MBS 很重要，因为当利率变化时，预期未来提前还款金额会变化，这样未来提前还款的时间会变化，因此未来现金流的时间同样会变化.

为了估计条件对冲比率，我们经常使用一个结构模型（同样需要用基于模型的 MBS 估价方法）. 这有两个不利条件：第一，对期限结构如何随时间活动以及这些活动如何与提前还款相关联的合理解释没有达成一致观点. 模型价格会与这些合理或不合理的假设密切相关[31]. 第二，更微妙的是对参数值可能往往是从一个静态的视角来选择或估计[32]. 例如，经验提前还款模型经常根据房屋和利率因素的数据反映提前还款率. 但是任何精装的（well-documented）MBS 失败对冲意味着代表过去关系平均值的回归系数与描述当前期的变量因素不具相同关联. 换句话说，估计系数的静态样本内回归不是对未来样本外系数的准确估计值.

减少样本内和样本外估计值之间误差的一个很有用的方法可能是概率密度估计法.

多元密度估计法

多元密度估计（MDE）是估计一组变量的联合密度方法. 已知这些变量的组合和边际密度，可以计算其相关分布和条件时间，比如均值. 这个估计将 MBS 的预期收益和中期国债期货的收益相关联，在任何时点上都有其相关信息. 我们有 T 个观测值 $\mathbf{z}_1$，$\mathbf{z}_2$，…，$\mathbf{z}_T$，其中 $\mathbf{z}_t$ 是一个可能包括 MBS 和中期国债期货收益的 m 维向量，也可能包括几个描述经济状况的变量. 一个很流行的联合密度度量是 Parzen（固定窗口宽度）密度估计量：

$$\hat{f}(\mathbf{z}^*) = \frac{1}{Th^m}\sum_{t=1}^{T} K\left(\frac{\mathbf{z}^* - \mathbf{z}_t}{h}\right)$$

其中 $K(\cdot)$ 称为核（kernel）函数（其性质是积分后为 1），并且经常被选择作为一个密度函数，h 是窗口或平滑参数（帮助决定核函数有多紧），$\hat{f}(\mathbf{z}^*)$ 是 $\mathbf{z}^*$ 处概率密度的估计值. 在任何一点 $\mathbf{z}^*$ 上的密度都由一系列以 $\mathbf{z}_t$ 为中心的数据的平均密度来估计. 数据偏离估计点越远，其对估计密度的贡献越小. 因此，估计在数据高度聚集时最高，在数据分散时最低[33]. 常用的核函数是多元正态密度函数：

$$K(\mathbf{z}) = \frac{1}{(2\pi)^{m/2}} e^{-\frac{1}{2}\mathbf{z}'\mathbf{z}}$$

令 $\mathbf{z}_t = (R_{t+1}^{mbs}, R_{t+1}^{TN}, \mathbf{x}_t)$，其中 R_{t+1}^{mbs} 和 R_{t+1}^{TN} 分别为 t 到 $t+1$ 期的 MBS 和中期国债期货的一期收益，x_t 是 t 时刻的（$m-2$）维因素向量. 我们因此可以得到条件均值 $E[R_{t+1}^{mbs} \mid R_{t+1}^{TN}, x_t]$，即

当前经济状况下，已知国债收益波动的 MBS 预期收益，由 x_t 表示当前经济状况. 特别地，

$$E[R_{t+1}^{mbs} \mid R_{t+1}^{TN}, \boldsymbol{x}_t] = \int R_{t+1}^{mbs} \frac{f(R_{t+1}^{mbs}, R_{t+1}^{TN}, \boldsymbol{x}_t)}{f_1(R_{t+1}^{TN}, \boldsymbol{x}_t)} \mathrm{d}R^{mbs}$$

$$= \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs} K_1^{t-i}(\cdot,\cdot)}{\sum_{i=1}^{t} K_i^{t-i}(\cdot,\cdot)} \tag{3.13}$$

其中 $K_1^{t-i}(\cdot,\cdot) = K_1((R_{t+1-i}^{TN} - R_{t+1}^{TN})/h^{TN}, (\boldsymbol{x}_{t-i} - \boldsymbol{x}_t)/h)$.

$K_1(\cdot,\cdot)$ 是边际密度，$\int K(z)\mathrm{d}R^{mbs}$ 也是一个多元正态密度. 公式（3.13）的预期收益仅是过去收益的加权平均，其中权重取决于条件变量水平相对于过去水平状况.

$E[R_{t+1}^{mbs} \mid R_{t+1}^{TN}, \boldsymbol{x}_t]$，对冲比例可以通过估计在现有信息 x_t 条件下 MBS 的收益变化关于中期国债期货收益变化的函数来形成. 即

$$\frac{\partial E[R_{t+1}^{mbs} \mid R_{t+1}^{TN}, \boldsymbol{x}_t]}{\partial R_{t+1}^{TN}} = \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs} \frac{\partial K^{t-i}(\cdot,\cdot)}{\partial R_{t+1}^{TN}}}{\sum_{i=1}^{t} K_1^{t-1}(\cdot,\cdot)} - \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs} K_1^{t-i}(\cdot,\cdot) \sum_{i=1}^{t} \frac{\partial K_1^{t-1}(\cdot,\cdot)}{\partial R_{t+1}^{TN}}}{\left[\sum_{i=1}^{t} K_1^{t-i}(\cdot,\cdot)\right]^2} \tag{3.14}$$

其中，$\frac{\partial K_1^{t-i}(\cdot,\cdot)}{\partial R_{t+1}^{TN}} = -\left[\frac{(R_{t+1-i}^{TN} - R_{t+1}^{TN})}{(h^{TN})^2}\right] K_1^{t-i}(\cdot,\cdot)$.

有几点需要指出. 第一，公式（3.14）给出了投资者的 MBS 持仓和中期国债期货之间的对冲比率公式. 比如，如果 $\frac{\partial E[R_{t+1}^{mbs} \mid R_{t+1}^{TN}, \boldsymbol{x}_t]}{\partial R_{t+1}^{TN}}$ 等于 0.5，则对于 MBS 的每 1 美元，投资者都应该做空 0.5 美元的中期国债期货. 第二，对冲利率是动态变化的，依赖于由 $\boldsymbol{x}_t$ 表示的当前经济形势. 比如，假设 $\boldsymbol{x}_t$ 是期限结构变量的 m－2 维向量. 不管这些变量是期限结构的水平、斜率还是曲率，对冲比率都会随其相应变化. 因此，中期国债期货的合适持仓量会随时间变化. 第三，对冲比率是关于中期国债期货的未知收益的函数. 如果 MBS 收益和中期国债期货收益总是线性关系，那么无论中期国债期货如何变化，合适的对冲比率总是一样. 如果是非线性关系，则投资者必须决定用哪种类型的中期国债来进行对冲. 比如，投资者可能想要在中期国债期货收益的条件均值附近对 MBS 进行对冲，因为很多潜在的中期国债期货位于这个区间范围内. 另一方面，亦可能是因为投资者考虑到中期国债期货收益分布的尾巴，所以调整对冲比率把利率和中期国债期货的潜在极端波动考虑进去. 第四，对冲利率是水平明确的. 与即时对冲比率相反，这个方法暗含的对冲比率直接反映相关水平上的 MBS 收益分布. 因此，每日、每周或每月水平上可能适合不同的对冲比率.

静态普通最小二乘法（OLS）的回归系数或对冲比率如下：

$$\beta = \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs} R_{t+1-i}^{TN} - T\mu_{mbs}\mu_{TN}}{\sum_{i=1}^{t} (R_{t+1-i}^{TN} - \mu_{TN})^2} \tag{3.15}$$

其中 $\mu_{mbs} = \frac{1}{T}\sum_{i=1}^{t} R_{t+1-i}^{mbs}$，$\mu_{TN} = \frac{1}{T}\sum_{i=1}^{t} R_{t+1-i}^{TN}$.

相反，动态对冲方法明确地将当前经济状况考虑进去. 公式（3.14）可以被重写为：

$$\frac{\partial E[R_{t+1}^{mbs} \mid R_{t+1}^{TN}, \boldsymbol{x}_t]}{\partial R_{t+1}^{TN}} = \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs}\left[\frac{(R_{t+1-i}^{TN} - R_{t+1}^{TN})}{(h^{TN})^2}\right] w_i(t) - \left[\sum_{i=1}^{t} R_{t+1-i}^{mbs} w_i(t)\right]}{\left[\sum_{i=1}^{t}\left(\frac{(R_{t+1-i}^{TN} - R_{t+1}^{TN})}{(h^{TN})^2}\right) w_i(t)\right]}. \qquad (3.16)$$

其中 $w_i(t) = \dfrac{K_1^{t-i}(\cdot,\cdot)}{\sum_{i=1}^{t} K_1^{t-i}(\cdot,\cdot)}$.

通过 MBS 和中期国债期货收益的过去数据组建公式（3.16）中的对冲比率，通过判断数据组（R_{t+1-i}^{TN}，$\boldsymbol{x}_{t-i}$）与选定的 R_{t+1}^{TN} 值和当前信息 x_t 的接近程度来给这些数据组的协同运动赋予不同的权重. 动态对冲比率在理论上与回归对冲相似，除非权重不再固定，而是依赖于当前信息. 如果当前信息 $\boldsymbol{x}_t$ 在分布上不接近于 $\boldsymbol{x}_{t-i}$，则数据组（R_{t+1-i}^{mbs}，R_{t+1-i}^{TN}）的权重 $w_i(t)$ 很小. 对冲比率根据当前经济状况进行调整. 比如，如果当前利率很高，但是期限结构是颠倒的，那么这种利率环境下的 MBS 和中期国债期货过去的协同运动会被赋予更高的权重.

Boudoukh、Richardson、Stanton 和 Whitelaw（1995）对 30 年期固定利率的 GNMA MBS（息票率为 8%、9%和 10%）及从 1987 年 1 月到 1994 年 5 月的中期国债期货数据运用了这个方法. GNMA 价格代表交易商对息票率为 X%的基差待定[34]的远期交易的 GNMA 的报价. 对于样本外分析，他们的研究表明动态对冲方法比静态回归方法更好. 比如，在对冲息票率为 10%的 GNMA 的周收益时，动态方法将 GNMA 收益波动从 41 个基点降低到 24 个基点，然而静态方法仅仅控制了剩下的 29 基点. 另外，动态对冲时只有一个基点的收益波动归因于美国国债风险，而静态对冲时则有 14 个基点的收益波动归因于利率风险.

Boudoukh、Richardson、Stanton 和 Whitelaw（1995）的结果如表 3.26 所示，对 GNMA TBA 的未对冲收益和两种不同方法下的对冲收益的均值、波动率和自相关进行比较. 这些方法涉及用中期国债期货来对冲 GNMA，得到对冲收益 $R_{t+1}^{mbs} - \beta_t + R_{t+1}^{TN}$，其中 R_{t+1}^{mbs} 和 R_{t+1}^{TN} 分别为 GNMA 和中期国债期货的样本外收益，对冲比率可用以下两种方法根据前 150 周的数据进行估计：(1) 根据 R_{t+1}^{mbs} 关于 R_{t+1}^{TN} 的回归来进行线性对冲；(2) 利用 t 时刻的 10 年期收益条件上的 R_{t+1}^{mbs} 和 R_{t+1}^{TN} 分布进行 MDE 对冲. 这个估计是在一个滚动基础上进行，并且覆盖了样本外的时间段，1989 年 12 月到 1994 年 5 月. 结果报告每周和每月重叠收益.

表 3.26

	GNMA8		GNMA9		GNMA10	
	1 周	4 周	1 周	4 周	1 周	4 周
未对冲						
均值（%）	0.078	0.326	0.077	0.316	0.069	0.286
波动率（%）	0.685	1.364	0.531	0.999	0.414	0.746
自相关系数（%）	0.011	0.138	−0.019	0.109	−0.045	0.082
线性对冲						
均值（%）	0.007	0.041	0.017	0.086	0.024	0.126
波动率（%）	0.318	0.599	0.309	0.573	0.286	0.524
自相关系数（%）	0.015	−0.107	0.012	0.021	0.060	0.039

（续）

	GNMA8		GNMA9		GNMA10	
	1 周	4 周	1 周	4 周	1 周	4 周
MDE 对冲						
均值（%）	0.006	0.123	0.020	0.178	0.027	0.189
波动率（%）	0.285	0.583	0.245	0.472	0.242	0.440
自相关系数（%）	0.011	−0.096	0.016	−0.083	0.036	−0.085

资料来源：Boudoukh，Richardson，Stanton，and Whitelaw（1995）.

图 3.10 表示用 10 年期中期国债期货来对冲息票率为 10%（上图）和 8%（下图）的 GNMA 每周收益的对冲比率.对冲比率是在 150 周的滚动基础上使用线性回归和 MDE 来估计的. MDE 对冲比率受 10 年期中期国债收益水平的限制.

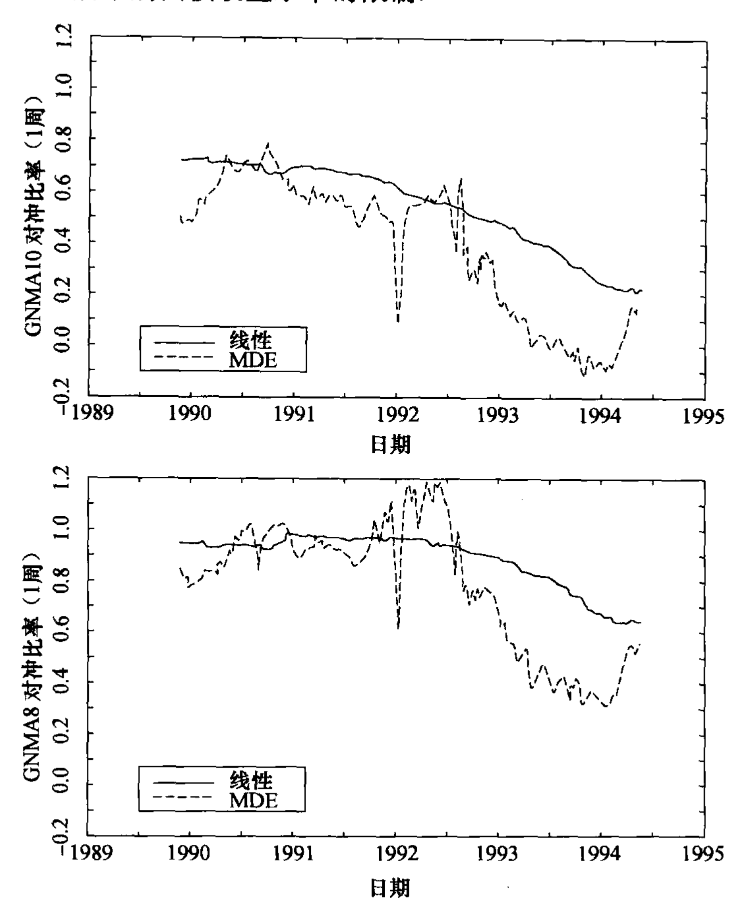
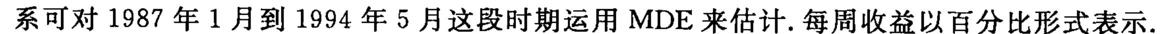


图 3.10 用 10 年期中期国债期货来对冲息票率为 10%（上图）和 8%（下图）的 GNMA 每周收益的对冲比率

资料来源：Boudoukh，Richardson，Stanton，and Whitelaw（1995）.

图 3.11 表示关于同期中期 10 年期中期国债期货收益的函数，息票率为 10%（上图）和 8%（下图）的 GNMA 的预期每周收益，受 10 年期中期国债收益的三个不同水平限制.这一关



系可对 1987 年 1 月到 1994 年 5 月这段时期运用 MDE 来估计. 每周收益以百分比形式表示.

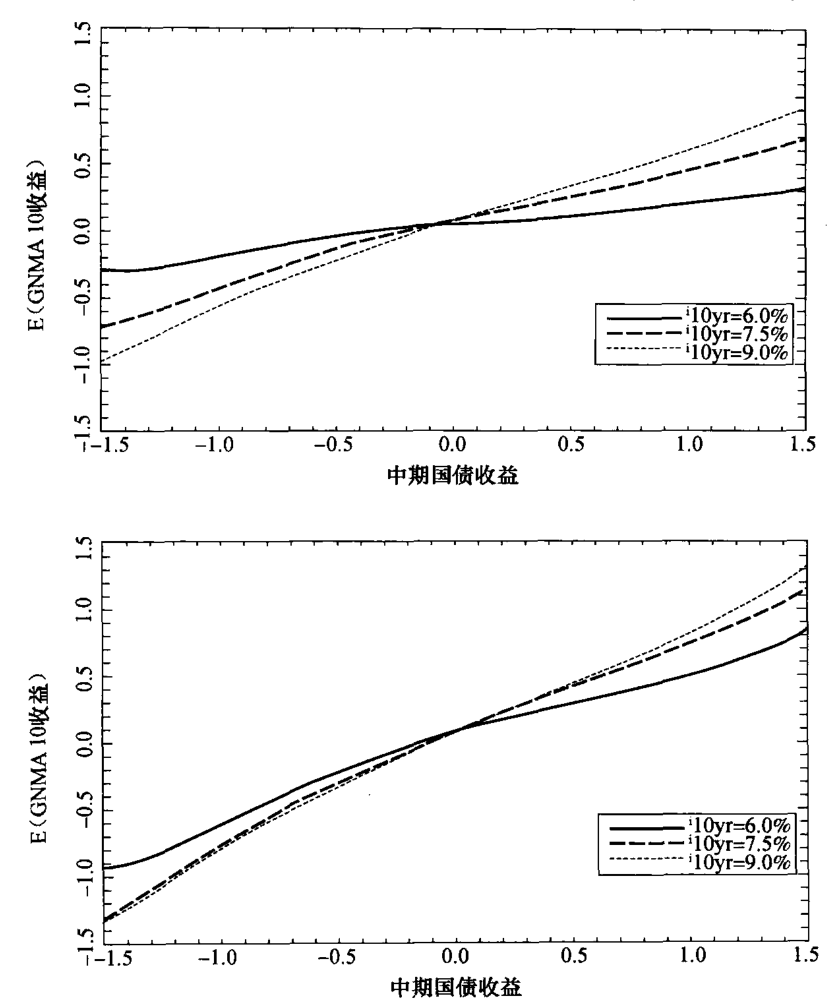


图 3.11　关于同期 10 年期国债期货收益的函数，息票率为 10%（上图）和 8%（下图）的 GNMA 的预期每周收益，受 10 年期国债收益的三个不同水平限制

资料来源：Boudoukh，Richardson，Stanton，and Whitelaw（1995）.

尾注

1. 本章参考泽维尔（Xavier）大学金融学教授 Stafford Johnson 博士的论文，见 http://www.academ. xu. edu/johnson/.
2. Obazee P(2002)，338～339 页．见《Interest Rate，Term Strueture and Valuation Modeling》中的 "Understanding the Building Blocks for OAS Models". Frank J. Fabozzi 编辑，Wiley & Sons 出版．

3. Id. 338～339.
4. Id. 338～339.
5. 在实际应用中，人们会运用比二项式模型（如 Hull-White，Black-Derman-Toy，Black-Karasinski 或 Cox-Ingersoll-Ross 的利率模型）更复杂的期限结构模型.
6. 在本章后面会给出 C++代码.
7. Hull J（1996），391 页.
8. Fabozzi F，Richard S，and Horwitz D（2002），445 页．《Interest Rate，Term Structure，and Valuation Modeling》中的 "Monte Carlo Simulation/OAS Approach to Valuing Residential Real Estate-Backed Securities"，Wileg & Sons 出版.
9. Id. 445～446.
10. Id. 445～446.
11. Id. 446.
12. Id. 453.
13. Obazee P（2002），315～344 页．见《Interest Rate，Term Structure，and Valuation Modeling》中的 "Understanding the Building Blocks for OAS Models"，Frank J Fabozzi 编辑，Wiley & Sons 出版.
14. Id. 317.
15. Id. 318.
16. Id. 319.
17. Bandic I，11 页.
18. 之前讨论了再融资动机和季度调整因子，现在给出基于 Richard&Roll（1989）的经验观测值的解析公式.
19. Dacidson/Herskovitz（1996）的公式.
20. Richard Roll（1989）中图 3 的每月参数.
21. Fabozzi F，Richard S，and Horwitz D(2002)，459 页．《Interest Rate，Term Structure，and Valuation Modeling》中的 "Monte Carlo Simulation/OAS Approach to Valuing Residential Real Estate-Backed Securities"，Wiley & Sons 出版.
22. Id. 454.
23. Fabozzi F，Richard S，and Horwitz D（2002），454 页．
24. Id. 455.
25. 单因子模型作为抵押贷款率函数，可以通过对 MBS 估价时将提前还款行为的多样性考虑进去来描述提前还款的衰减效应. 相反，三因子模型是基于离散时间、无套利定价理论，将提前还款行为与现金流形式联系起来，这里提前还款行为是缘于再融资（利率变化引起）和不断上涨的房价.（见 Kariya and Kabayashi，2000，Kariya，Ushiyama，and Pliska，2002.）
26. Boudoukh，Richardson，Stanton，and Whitelaw（1995），1 页.
27. Id. 1.

28. Id. 1.
29. 关于静态对冲问题的一些讨论，可参见 Breeden(1991) 和 Breeden and Giarla(1992). 尤其是对于线性回归对冲，Batlin(1987) 讨论了提前还款期权对 MBS 与中期国债期货的对冲比率的影响.
30. Davidson and Herkowitz(1992) 分析了 MBS 实际定价的各种理论方法，并详细讨论了每种方法的优缺点. 特别针对 MBS 的对冲，Roberts(1987) 主要分析了基于模型的 MBS 估价方法.
31. Id.
32. Id. 2.
33. Id. 3.
34. TBA 市场主要由发放未被组合的住房抵押贷款的发起人占领. 然而，交易也涉及未指定基准的已经存在的组合. 这意味着在交易达成一致时，即将交割的抵押贷款组合的特征（如组合寿命、提前还款历史等）由交易商自行决定. 但是，一旦发行了新的具有要求息票率的抵押贷款，这些组合很可能被交割，因为在具有样本期特征的利率环境下，季度调整的组合更有价值. 因此，GNMA TBA 被看做是关于新发行的通用证券的远期合约.

第 4 章　债务抵押债券

本章将讨论债务抵押债券（Collateralized Debt Obligation，CDO）这种结构化金融产品. 4.1 节将讨论债务抵押债券的结构，4.2 节讨论合成债务抵押债券，4.3 节回顾一下用债务抵押债券进行资产负债表风险管理的重要性. 4.4 节讨论资产组合的违约亏损分布，4.5 节着重关注债务抵押债券的股权份额. 4.6 节介绍债务抵押债券的份额定价方法. 4.7 节研究一下债务抵押债券的份额定价公式. 4.8 节介绍债务抵押债券的仿真定价算法. 4.9 节提供一种用 Matlab 实现的债务抵押债券定价方法. 4.10 节介绍如何在 C++ 中实现其定价并给出一个实例. 4.11 节讨论债务抵押债券的债务抵押债券（CDO^2）. 4.12 节讨论债务抵押债券和 CDO^2 的快速亏损计算方法——该方法使用了条件常态近似法，还提供了一种基于 Matlab 计算债务抵押债券份额亏损快速算法.

4.1　债务抵押债券的结构

债务抵押债券是一种流动性较差、信贷风险较高的多样化资产组合. 这些资产可以是高收益债券，如债券抵押债券（Collateralized Bond Obligation，CBO），也可以是银行杠杆贷款，如贷款抵押债券（CLO）. 债务抵押债券也可以包括结构化固定收益证券、不良债权、新兴市场债权以及商业房地产相关债权. 债务抵押债券由一个特定交易机构（SPV）管理，根据不同的风险收益特性以及优先权，债务抵押债券被构造并打包成为多个不同份额. 针对各份额优先权的不同，债务抵押债券运用不同的债务工具对其分配本金和利息的支付权. 图 4.1 为一个典型的债务抵押债券结构.

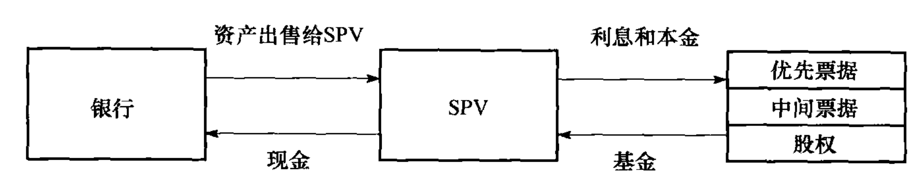


图 4.1　CDO 结构

资料来源：Picone D(2001)，15.

优先票据（senior note）比中间票据（mezzanine note）和低级票据有优先支付权. 因此，在每个支付日，本金和利息先向优先票据份额支付，再向中间票据份额支付，随后向低级票据支付. 股权份额则可以获得剩余的现金收益. 股权份额称为抵押资产组合中的“先损头寸”，因为这一份额承受了组合中第一块钱的损失的风险. 根据债务抵押债券利用相关资产产生现金收益从而来履行债务的能力的不同，评级机构对其信用进行评级. 债务偿还“取决于抵押资产多样性、质量方针、组织形式、生产结构（信用强化措施、流动性保护措施）”.[1] 当某一份额在债务抵押债券中的优先级降低时，风险的级别就会增高. 股权份额持有者所承受的风险最高，但是他们在不可赎回期过后有权赎回. 不可赎回期往往是 3～5 年.

典型的债务抵押债券包括构建阶段（在这一时期中抵押资产组合形成）、再投资阶段

（这一阶段中对资产组合进行主动管理运作）以及契约解除期（这一时期按照份额的优先级用所得收益对债务进行偿还）. 在再投资阶段，"权益级（equity class）分布包括整个资产组合的多余利息，再减掉在支付了债务利息和其他各项费用后剩余的抵押物收益". 在偿还期间，随着通过被抵押资产的本金带来的收益按照优先级别对债务进行偿还，额外利息的支付也在逐步减少. 当所有债务类债务付清之后，如果权益级没有被赎回，那么剩余的本金都归入权益级中[2].

把债券作为抵押物的债务抵押债券称为*债券抵押债券*（CBO），而把银行贷款作为抵押物的债务抵押债券称为*贷款抵押债券*（CLO）. 在这两种契约中，由信托基金或特定交易机构发行票据，票据的抵押物相应为债券或银行贷款. 通常这些票据以多份额的形式向投资者发行. 这些份额根据它们本身信贷风险特性的不同，最高被评为 AAA，最低为未评级的权益级.

根据份额优先等级不同，相关资产组合的亏损风险被划分到不同的份额之中. 亏损最先影响到股权份额（"先损"份额），随后是中间份额，接着是优先份额，最后是超优先份额. 中间份额对于商业周期的敏感度很高，通常被评为投资级，而超优先份额的等级比 AAA 级更高.[3] 根据投资者的风险厌恶特性不同，他们买入不同等级份额的债务抵押债券，并向债务抵押债券发行商售出信贷保护. 相应地，发行商则以单名信用违约掉期的形式出售资产组合的信贷保护，以对冲自己的风险. 整个套期保值交易的双方是债务抵押债券的原始发行人和该契约的投资者，而发行机构则只在其中充当中间人的角色.[4]

图 4.2 给出了一个资本结构的范例，其中高收益债券充当了这个债务抵押债券的抵押资产.

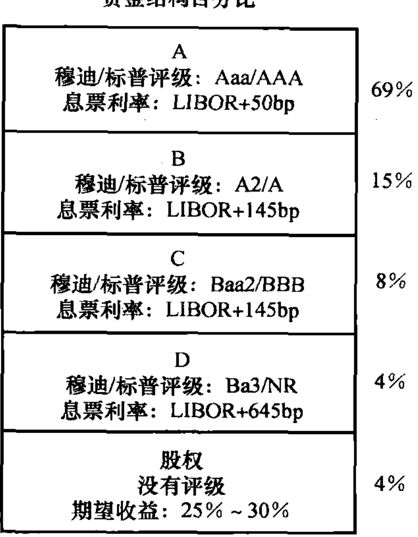


图 4.2 CDO 抵押品结构

资料来源：Picone D(2001).

大部分债务抵押债券可以分为两类：套利类和资产负债表交易类.[5] 套利类债务抵押债券可以继续分为现金流型债务抵押债券和市场价值型债务抵押债券. 而资产负债表类债务抵押债券包含资产负债表类现金流型债务抵押债券. 本章的内容直接依照 Picone（2001）所做的工作来进行. Picone 解释了各种债务抵押债券（还可参见 Lehnert、Altrock、Rachev、Trück 和 Wilch [2005]）.

4.1.1 现金流型债务抵押债券

现金流型债务抵押债券的特点是它的抵押资产组合不受债券管理者的交易管理行为影响. 与本金和利息提前偿还有关的不确定性是由抵押资产的违约时间和违约数量决定的. 各个份额票据的资本回收直接同抵押资产组合带来的现金流挂钩. 违约损失是资本回收的主要风险.

4.1.2 市场价值型债务抵押债券

市场价值型债务抵押债券的特点是其各份额的绩效主要是其市场绩效. 也就是说，所有被

抵押的证券都以很高的频率紧盯市场.市场价值型债务抵押债券对债务抵押债券经理的表现有杠杆作用.潜在的 CDO 投资者需要对资产组合管理者的能力、工作环境的组织结构以及自己是否适合于杠杆管理这一风格都做一番仔细的评估，这个调查是有必要的.

4.1.3 资产负债表类现金流型债务抵押债券

资产负债表类债务抵押债券的目的是为了资本救济，被证券化的资产是低收益债务工具.资本救济有助于降低融资成本或者增加股权收益，这是因为该债券将管理资本高的资产从资产负债表中去除了.这类交易依赖于高回收率银行贷款担保资产的质量.由于资产负债表类现金流型债务抵押债券票面利率相对较小，因此和相应的套利型债务抵押债券相比，息差缓冲更小.然而，由于它们具有较高的质量，所具有的从属性就相对较小.大部分情况下，被出售的资产是由贷款担保的资产组合.通常，一个资产负债表债务抵押债券都会很大，因为该交易必须对寻求资本救济的公司的每股收益产生一定影响.

4.1.4 套利类债务抵押债券

套利类债务抵押债券的目的在于抓住存在于高收益的抵押资产和高回报率票据之间的利差的套利机会.其方法就是构造一个融资成本比票据回报率更低的抵押资产.大部分套利交易都是私下进行的，规模和数量都难以与现金流型的交易相比.

4.1.5 套利类市场价值型债务抵押债券

和资产负债表债务抵押债券的资产组合中贷款没有频繁的交易不同，为了利用被发现的价格优势，套利债务抵押债券经理会进行大量交易.这类债务抵押债券依赖于证券化资产的市场价值，这一价值每天更新.每一种在资本市场中进行交易的具有一定预期波动性的证券，都可以归为这类债务抵押债券.实际上，主要应当考虑的是抵押资产的价格波动性.债务抵押债券经理赚取高收益的能力是非常重要的.债务抵押债券经理对资产的使用具有很高的灵活性，循环贷款期间债券经理可以选择增加或减少融资，从而改变债务抵押债券的杠杆.

4.1.6 套利类现金流型债务抵押债券

抵押资产往往以市场价购买，属于流动证券.大部分抵押资产是债券.但是过去，曾经有可以交易的合成贷款也在其列.作为套利交易，抵押资产可以通过将信贷风险和融资成本重新分配成更加多样化的资产组合来更经济地再融资.

4.1.7 现金流交易中的信用强化

现金流交易中的优先票据受其他次级别的份额、过量担保和超额利差（excess spread）保护.优先票据有对现金流的优先认领权，因此非优先票据的业绩受制于高级票据业绩.过量担保对最低担保价值施加两项覆盖测试：票面价值测试和利息覆盖测试，从而为优先票据提供了更进一步的保护.票面价值测试要求优先票据（然后还有其他各档票据）是抵押资产的一定百分比（如 110%）.票面价值测试同样适用于优先级较低的票据，如中间票据.在这一测试中，如果低于触发百分比，则表示测试失败.触发百分比一般选择一个较低的比率

(如 105%). 利息覆盖测试要求保证利息能够覆盖所有亏损并仍能支付给优先票据持有者. 这类信贷支持也称作超额利差.

4.1.8 市场价值交流中的信用强化：担保率和过量担保测试

在市场价值交易中，*担保率*（advance rate）是信用强化的一种主要形式. 担保率是指能够用来发行债务的市场率的最大百分比. 信用评级机构对不同的抵押物标识不同的担保率. 这一比率取决于抵押资产的收益率的波动性和资产在市场中的流动性. 收益波动率高、流动性差的资产的担保率较低. 表 4.1 列举了 Fitch 对不同资产设定担保率的例子. 比如说对于 AA 级的债务，可能需要存款证的 95%来作为抵押资产. 而如果要发行一个 BB 级的相同类型债券，可能需要市场价值的 100%作为抵押资产.

表 4.1 Fitch 担保率

资产类别	AA	A	BBB	BB	B
现金和现金等价物	100%	100%	100%	100%	100%
CD 和 CP	95%	95%	95%	100%	100%
优先银行抵押贷款	85%	90%	91%	93%	96%
BB 级高收益率债务	71%	80%	87%	90%	92%
<BB 级高收益率债务	69%	75%	85%	87%	89%
可转换债券	64%	70%	81%	85%	97%
可转换优先股	59%	65%	77%	83%	86%
夹层债务、抵押物、新兴市场	55%	60%	73%	80%	85%
股权、流动债务	40%	50%	73%	80%	85%

资料来源：Fitch.

对于市场价值交易来说，通常有多个过量担保测试. 为了说明这些测试的原理，请先考虑一下表 4.2 的抵押物和债务结构.

表 4.2 CDO 市场价值交易价格

抵押物			票　据			
资　产	市场价值（百万欧元）	%	份　额	评　级	面值（百万欧元）	%
CD 和 CP	£230	46	优先信贷	AA	£175	35
BB 级高收益率债务	£225	45	优先票据	AA	£200	40
可转换债券	£10	2	中间票据	BBB	£50	10
夹层债务	£25	5	次级票据	B	£25	5
股权	£10	2	股权	NR	£50	10
总计	£500	100%	总计		£500	100%

资料来源：Picone D，5.

在给予 AA 级的担保率以后，如表 4.3 所示，预付款总额超过了总的 AA 级中债务数量 2800 万英镑，这个余额也称为借贷剩余（borrowing amount surplus）. 这就是 AA 级债务在没有违反过量担保测试（OC 测试）前所能承受的市场价值损失.

表 4.3 AA 等级债务 OC 测试

抵押物				
资　产	市场价值（百万欧元）	百 分 比	AA 评级比率	预付款总额（百万欧元）
CD 和 CP	£230	46%	95%	£219
BB 级高收益率债务	£225	45%	71%	£160
可转换债券	£10	2%	64%	£6
夹层债务	£25	5%	55%	£14
股权	£10	2%	40%	£4
总计	£500	100%		£403

AA 级优先债务面值（优先信贷+优先票据）：£375
借贷剩余：£28

资料来源：Picone D，6.

表 4.4 和表 4.5 分别列举了 AA+BBB 级债务和 AA+BBB+B 级债务的借贷剩余. 与表 4.3 一样，他们的借贷剩余分别为 2 300 万欧元和 2 500 万欧元，而这些正是他们在违反过量担保测试前所能够承受的最大市场价值损失.

表 4.4 AA+BBB 级债务 OC 测试

抵押物				
资产	市场价值（百万欧元）	百 分 比	BBB 评级比率	预付款总额（百万欧元）
CD 和 CP	£230	46%	95%	£219
BB 级高收益率债务	£225	45%	87%	£196
可转换债券	£10	2%	81%	£8
夹层债务	£25	5%	73%	£18
股权	£10	2%	73%	£7
总计	£500	100%		£448

AA+BBB 级优先债务面值：£425
借贷剩余：£23

资料来源：Picone D，6.

表 4.5 AA+BBB+B 级债务 OC 测试

抵押物				
资　产	市场价值（百万欧元）	百 分 比	B 评级比率	预付款总额（百万欧元）
CD 和 CP	£230	46%	100%	£230
BB 级高收益率债务	£225	45%	92%	£207
可转换债券	£10	2%	87%	£9
夹层债务	£25	5%	85%	£21
股权	£10	2%	85%	£9
总计	£500	100%		£475

AA+BBB+B 级优先债务面值：£450
借贷剩余：£25

资料来源：Picone D，6.

抵押资产经理必须确保在标的物价格波动时，没有违反市场价值测试. 违反过量担保测试是非常严重的，而当它发生时，抵押资产经理有两种方法通过一个 2～10 天的回收期来进行补偿：

1. 卖出一份担保率较低的证券而买入一份或多份担保率更高的证券.
2. 卖出一份担保率较低的证券并偿还有更多优先票据的债务.

当违反过量担保测试情况不很严重时，一般更倾向于使用第一种方法. 第二种方法是一个非常极端的回收方式. 当抵押资产经理无法满足过量担保测试的要求时，债务人有权接管基金并清偿债务，以防止债务违约的发生.

4.1.9 最小净值测试

最小净值测试通过构建一个股权缓冲，也被用来担保优先票据持有者的信贷风险. 测试的方法就是确保资产市场价值（MAV）减去债务票据能够不小于股权的面值乘以一个百分比.

$$\text{资产市场价值} - \text{债务票据} \geqslant \% * \text{股权面值}$$

万一无法满足测试的要求，经理有一个回收期来使债务抵押债券能够重新满足测试要求. 有两种方法：(1) 赎回部分或全部优先票据；(2) 通过卖出部分资产来提高资本收益率. 第二种方式一般更受欢迎，因为经理不会减少交易的杠杆. 如果抵押资产经理不能满足最小净值测试的要求，并且有债务违约的情况发生，那么债务人有权执行交易.

图 4.3 展示了一个 CBO 的实际结构以及 2004 年 3 月 22 日起开始营业的 Iccrea Banca Spa 的条款说明书. 该 CBO 被意大利合作银行（Italian Co-operative Bank）的浮动利率债券证券化. 表 4.6 展示了票据结构.

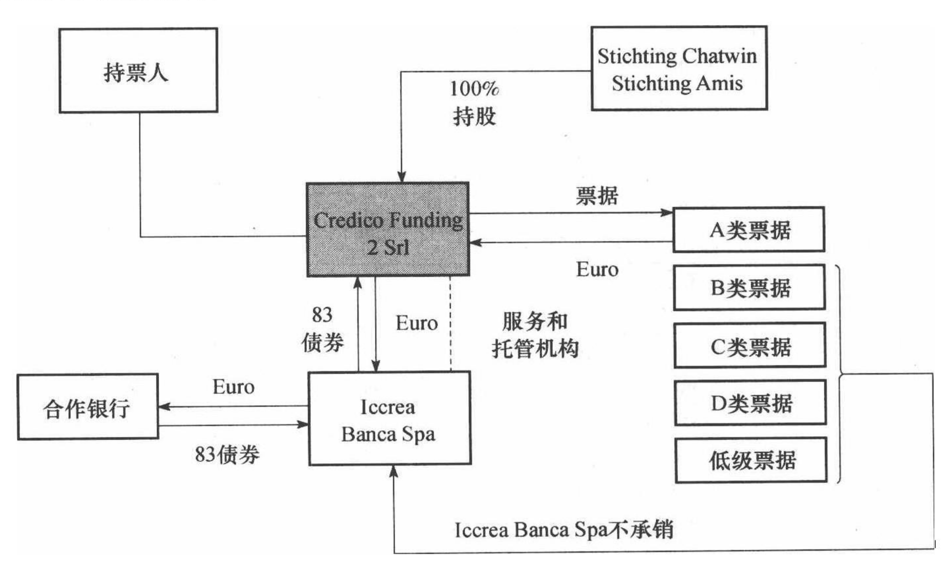


图 4.3

资料来源：意大利合作银行 CBO.

表 4.6 票据结构

等级	初始本金余额	等级规模	评级	欧元同业拆借利率	平均寿命	期望到期日	标准到期日
A	EUR[1.131]mn	[87.0]%	AAA	+25bp	6.0 年	[Mar 2010]	[Mar 2012]
B	EUR[39]mn	[3.0]%	AA	+40bp	6.0 年	[Mar 2010]	[Mar 2012]
B	EUR[65]mn	[5.0]%	A	+80bp	6.0 年	[Mar 2010]	[Mar 2012]
B	EUR[26]mn	[2.0]%	BBB	+120bp	6.0 年	[Mar 2010]	[Mar 2012]
B	EUR[39]mn	[3.0]%	NR	+300bp	N/A	N/A	N/A

4.1.10 交易特征

Credico Funding 2 Srl（“Credico 2”）是意大利合作银行（BCC）的第二项证券化资产.第一版在 2002 年 3 月完成.

一组由 83 个优先的未评级、未保险、未上市的浮动利率债券组成的债券组合对 Credico 进行担保.这个债券组合发行于 2003 年 3 月 29 号，由 462 家 BCC 中的 83 家发行.2004 年 4 月 15 号，Credico Funding 2 Srl 以总价 12 亿买入该组合.债券组合中的债券的最晚到期日在 2010 年 3 月.

Credico 2 会发行 6 个级别的票据，分别被评定为 AAA、AA、A 和 BBB.中间票据和低级票据不由 Iccrea 进行承销.

根据意大利银行的确认，所有类别票据在意大利承担 20%的风险.

只有 A 类票据在欧洲司法范围（如德国、英国、爱尔兰、比利时卢森堡荷兰经济体、法国、西班牙和荷兰）内承担 20%的风险.表 4.7 展示了交易的相关方.

表 4.7 交易各方

发行机构	Credico Funding 2 Srl，基于意大利证券法 130/99 所建立的一种特殊的投资工具
卖方	ICCREA Banca S. p. A.（“ICCREA”）
持票人	银行家信托有限公司
服务商/托管人/信贷银行	ICCREA Banca S. p. A.
Stichtingen	Stichting Chatwin 和 Stichting Amis 是基于荷兰相关法律建立起来的
本金计算支付代理/代理银行	德意志银行，AG，伦敦
意大利支付代理/开户行	德意志银行 S. p. A.
卢森堡支付代理	德意志银行卢森堡 SA
服务管理供应商	SPV 管理有限公司
企业服务供应商	德勤会计师事务所
结算（交割）	Monte Titoli/欧洲结算系统/Clearstream
上市	卢森堡证券交易所
安排商	ICCREA Banca S. p. A.
顾问	Société Générale
评级机构	惠誉评级、穆迪和标普
联合承销商	美国银行/CAI/Société Générale
发行日期	2004 年 4 月 19 日
截止日期	2004 年 4 月 26 日

表 4.8 提供了债务抵押债券（CDO）交易细节信息.

表 4.8 交易结构

抵押物	由未经评级和经标普评级后共同持有的、发行于 2004 年 3 月 29 日的无担保和未上市的流动债券（债券投资组合）. 购买这些将花费 12 亿欧元资金，并将于 2004 年 4 月 15 日转换为 Credico 2. 此债券投资组合中的每一种债券的最后到期日为 2010 年 3 月
利率	A 等级票据利息为 3 个月欧元同业拆借利率加上 0.25% B 等级票据利息为 3 个月欧元同业拆借利率加上 0.4% C 等级票据利息为 3 个月欧元同业拆借利率加上 0.8% D 等级票据利息为 3 个月欧元同业拆借利率加上 1.2% E 等级票据利息为 3 个月欧元同业拆借利率加上 3%
利息支付日期	相关债券季度欠款的支付以 3 个月的 Euribor+[•]为基准，同时票据的支付将遵循同一支付投资组合，这样可以自然地进行对冲. 首期票据利息支付日期是[•]（首期短期息票），随后每年进行的首期短期息票支付日期是[•]，[•]，[•]
预期赎回日期	利息支付日的前半期在[•]；利息支付日期的后半期紧随累计损失事件的发生，利息支付日在[•]
法定到期日	[]2013
赎回	连续
加强信贷措施/变现能力	按 B、C、D、E 等级划分，首期债券投资组合总额（从这个交易前 15 个月的超额利差中积累，并且如果超过交易期限，则被补充）的流动储备资金提高至 0.22%. 此外，在整个交易进程中，利息支付日期的当天，以 3 个月 EURIBOR 为基准加减以作为额外的储备账户来弥补债券投资组合中的差额
面额	100 000 欧元
票据管制法律	意大利
累计亏损事件	当前一募款期本金差额总额低于首期资金总额的 6%时，累计亏损事件发生
本金差额	违约债券中未偿还本金和由债券投资组合出售中衍生出的相关的偿还之间的差
储备金额	首期债券投资组合总额的 0.22%（从这个交易前 15 个月的超额利差积累的）
额外储备金	在整个交易进程中，低等级的票据将被冻结，并贷记入额外储备金账户，直至到期日释放
合格的投资项目	由机构担保，在到期日提供固定本金总额的任何欧元面值优先债务证券或其他债务，负债项评级不低于评级机构所要求的最低范围
发行商可用资金	**可用资金收益** 1）在符合交易文件条件下进行赔偿 2）债券投资组合各方面利息的征收 3）基于利息信贷账户的差额结余 4）任何利息的增加或信贷账户的变化 5）任何符合投资的流动资金总额减去本金资金投资总额和储备资金投资总额 **可用本金资金** 1）债券投资组合中各本金资金的征收 2）从出售的违约债券（不包括预期赎回日）中获得的补偿；纯本金资金投资总额 3）基于本金信贷账户的差额结余
预实施优先支付	**可用收益资金**将按照下列注解被应用于每个利息支付日期： 1）到期时由票据持有人接受支付 2）a）到期时由其他的交易双方接受支付；b）另外其余支付是为了维护全体发行商；c）支出储备账户需保留 10 000 欧元 3）A 级票据到期时利息总额 4）为填平 A 级票据的本金差额，在赎回日之后（包括当天）记入贷方的本金账户 5）遵守承诺书规定，在到期时归还信贷银行 6）B 级票据到期时的利息 7）为填平 B 级票据的本金差额，在赎回日之后（包括当天）记入贷方的本金账户 8）C 级票据到期时的利息

（续）

	9）为填平 C 级票据的本金差额，在赎回日之后（包括当天）记入贷方的本金账户 10）D 级票据到期时的利息 11）为填平 D 级票据的本金差额，在赎回日之后（包括当天）记入贷方的本金账户 12）为填平 E 级次级票据的本金差额，在赎回日之后（包括当天）记入贷方的本金账户 13）贷记人储备资金账户直至填平储备总额 14）依据转让协议支付到期款项给 Iccrea 15）在到期日之前把 E 级票据的所有利息总额贷记入其他储备资金账户 16）到期日当天，把所有 E 级次级票据的利息总额贷记入其他储备资金账户 17）其他的都贷记入本金账户 如果在利息支付日当天，没有足额的可用收益资金可以用来支付，直到 A 级票据全部赎回前，发行商应该于利息支付日当天，把储备金账户与其余储备资金账户的差额进行填补
可用本金资金	在预期到期日之前，可用本金资金将不会用作任何票据的支付，取而代之的是进行合理的投资
托管到期	从预期到期日开始，在每个利息支付日，发行商可将可用本金资金用于以下的行为： 1）全额赎回 A 级票据本金 2）全额赎回 B 级票据本金 3）全额赎回 C 级票据本金 4）全额赎回 D 级票据本金 5）值得全额赎回 E 级初级票据的本金 6）第 1 项至第 7 项，除第 4、7、9、11、13 项之外，当可用收益资金不够时，可采取强制支付政策 7）保留本金账户内的剩余资金
本金差额	本金差额只有在追偿收付后才会增加，且立刻可以计算得出 这些本金差额应该被分配： 1）把初级票据记入未偿付本金总额 2）对于 D 级票据，如果本金差额高于 E 级票据未偿付的本金总额，则记入 D 级票据的未偿付本金总额 3）对于 C 级票据，如果本金差额高于 D 和 E 级票据未偿付的本金总额，则记入 C 级票据的未偿付本金总额 4）对于 B 级票据，如果本金差额高于 C、D 和 E 级票据未偿付的本金总额，则记入 B 级票据的未偿付本金总额 5）对于 A 级票据，如果本金差额高于 B、C、D 和 E 级票据未偿付的本金总额，则记入 A 级票据的未偿付本金总额
可选择性的赎回权	如果发行商在任何时间点都能清偿票据持有人，则：1）发行商将被要求减免或暂缓在税收、关税、费用方面的本金利息；2）由于法律政策的改变，发行商将在票据债务方面或交易文件方面不具有法律约束，发行商可在任何一个时点，当支付日利息下跌时，将票据赎回.对于期权，可在最初的 18 个月交易日内进行赎回

4.2 合成债务抵押债券

合成抵押债券使用信用违约掉期（Credit Default Swap，CDS）而不是债券或者贷款来作为标的物投资组合.经过这样的合成，这个债务抵押债券结构有了更大的灵活性.通过 CDS，信贷风险从债券发放者的资产负债表上转移到了特定交易机构上，而标的物资产的所有权仍然属于债券发行人.和现金流债务抵押债券相反，合成债务抵押债券不需要初始投资，所有份额的票据都可能有正有负，而现金流债务抵押契约需要一笔初始投入.[6] 此外，与债券抵押债券和贷款抵押债券不同，合成抵押债券的各份额不一定都要以票据的形式售出.假设利率不变，如果再加上一个和当前份额本金剩余相等的现金流，那么现金流债务抵押债券的一个份额就同合成抵押债券的一个份

额相等.[7] 合成抵押债券同样也可以通过纯衍生品交易的方式来寻求保护.这种结构不需要像大部分传统现金流债务抵押债券那样涉及法律上十分复杂的贷款转移，而资产实际上是被转移到了特定交易机构那里.这个结构同样可以免受标的物为贷款或债券时要进行规范说明这个特点的影响.对于信贷风险方面，合成债务抵押债券更倾向专注于投资类或高收益类之中的一种，尽管合成债务抵押债券交易的标的物资产最近包括了住房抵押证券、商业抵押证券、债务抵押债券的份额以及其他结构性金融证券.

第一个合成债务抵押债券交易于 1997 年在银行间完成.[9] 银行资产负债表交易的动机要么是有对冲信贷风险的欲望，要么是有减少管理成本的想法，要么是两者都有.[10] 自从这类早期的合成债务抵押债券交易之后，用来组合合成债务抵押债券结构的技术被用来构造有风险-收益信息的债务抵押债券份额，这类结构受到投资者的欢迎.后来的这些交易被称为套利交易（arbitrage deals)，因为它们的动机来自于信贷产品投资者的需求而不是银行的需求.[11]在合成 CDO 市场出现的早期（1997—1999 年)，几乎所有合成债务抵押债券交易的动机都是为了银行资产负债表考虑的.然而套利交易的增长速度非常快，现在大部分市场交易都属于套利交易.同样，在早期的合成债务抵押债券市场，交易都包含了一整套完整的份额类别（股权类、中间类和优先类)，这些份额的票面价值总和相当于整个资产组合的票面价值.现在，大部分合成债务抵押债券交易由一个所谓的单份额债务抵押债券组成，也就是说，只有债务抵押债券结构中的一个份额被出售.[12]

图 4.4 列举了一个从 2002 年 1 月到 2004 年 2 月发行的一个合成债务抵押债券的票面价值.票面价值根据抵押物资产类型不同被进一步区分.在这段时间内，总发行量增长迅速，平均月发行量为 370 亿美元.[13]投资级别的公司债务是最常见的抵押物资产类型.结构融资是第二大类的抵押物资产并且现在越来越重要.这源于公司债券息差使得公司承担公司信贷风险对投资者的吸引力减小.高收益合成债务抵押债券的相对较小份额反映了高收益的单一品种信用违约掉期市场缺乏流动性这一情况.[14]

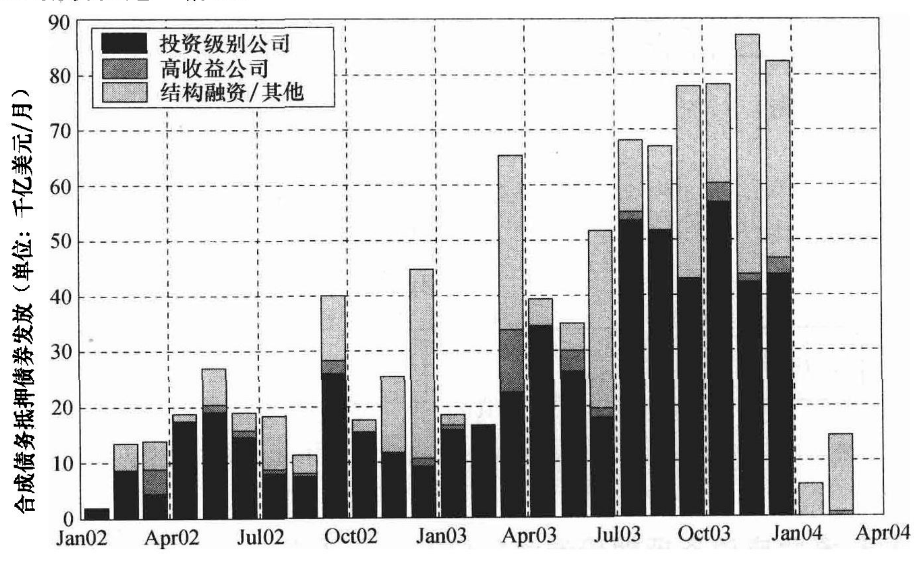


图 4.4 两种合成债务抵押债券

资料来源：Gibson M(2004)，2.

4.2.1 全额融资的合成债务抵押债券

对于一个全额融资的合成债务抵押债券（CDO），特定交易机构发行了大约相当于 100% 的抵押资产组合的票据. 而这些票据的收益投资于用作抵押物的优质证券，这些优质证券的风险权重为 0%. 为了能够对冲贷款资产组合中承受的风险，债券发行银行选择进入一个信用违约掉期（CDS）合约，该合约要么涉及同一个特定交易机构，要么涉及一个 OECD 银行. 通过这些信用违约掉期合约，贷款发放者相当于购买了一个信贷保护，这个保护的保费是基于债务方的信用等级的. 保费被加入利息当中归票据投资者所有. 通常，在一个融资合成债务抵押债券的份额中，CDO 投资者在交易之初支付该份额的票面价值，而任何违约都会引发本金亏损直至本金耗尽. 在整个交易中，投资者获得的是 LIBOR 加上反映该份额风险的息差. 投资者的资金被放入抵押物账户并投资于低风险证券（政府债务或 AAA 级债务）.[15]

发行者获得的股权承受 100% 的风险权重. 因此，如同图 4.4 中的结构，发行银行可以释放 6% 的管理资本. 额外需要的管理资本取决于结构中是否涉及 OECD 银行. 如果信用违约掉期直接与特定交易机构进行交易（如图 4.5 所示），且票据收益投资于 0% 风险权重的资产，那么此交易没有增加任何管理资本. 如果信用违约掉期通过 OECD 银行进行交易（如图 4.6 所示），那么该合约所需的管理资本就是掉期合约票面价值的 1.6%（20%×8%）. 如果违约掉期合约的票面价值和相关资产组合的价值相等，那么该交易总的管理资本费用就是 3.6%.

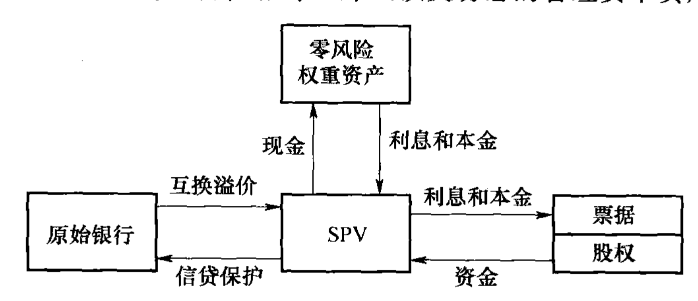


图 4.5 全额融资的合成债务抵押债券与信用违约掉期和 SPV

资料来源：Picone D(2001).

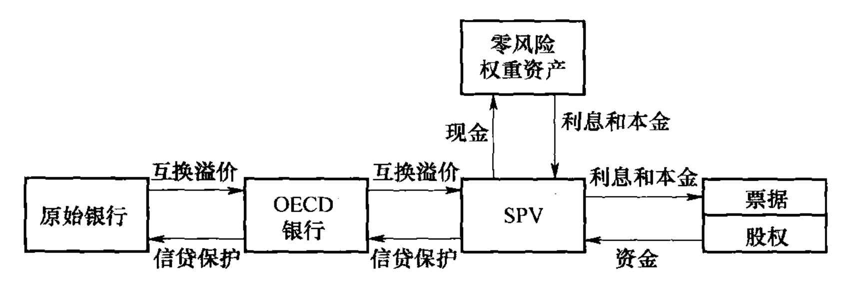


图 4.6 全额融资的合成债务抵押债券与通过 OECD 银行的信用违约掉期

资料来源：Picone D(2001).

4.2.2 部分融资合成债务抵押债券和融资型的合成债务抵押债券的未融资部分

由于管理资本的存在，一个全额融资的债务抵押债券并不能使原始发行银行获得最有效的

资本使用效率.一个全额融资的债务抵押债券一贷款抵押债券(CDO-CLO)有时候会成为一个相对昂贵的工具.然而,作为一项期限性的融资债务(term funding debt),CDO-CLO 结构确实更少暴露于息差可能变大的风险.一项更有效的资本配置方法是采用部分融资的债务抵押契约.部分融资的债务抵押债券在结构上同全额融资的情况非常类似.债券发行银行直接向 SPV(如图 4.4 所示)或者 OECD 银行(如图 4.5 所示)购买信贷保护.不同之处在于 SPV 发行的票面价值更少,这是因为银行所承诺的抵押担保资产更少.超级优先债券,作为未融资的部分,是这种结构的主要特点.超优先票据是非常优质的商业票据,它们不会有承担亏损的风险.发行银行与 OECD 银行就超优先部分进行一个 CDS 的交易(超级优先 CDS).同样,银行也能要么同 SPV 要么同 OECD 银行进行一个初级 CDS 的交易.

未融资份额同一个掉期很相像[16].交易之初没有现金流变化.投资者在资产组合中的债务违约对投资人所在份额造成影响(在先前的违约情况将次级份额消耗殆尽之后)时向合约另一方支付,而能够获得息差的利益.[17]由于未融资份额依赖于投资者在未来向该合约继续投入现金的能力和意愿,这实际上就造成了合约另一方的信贷风险,而这种风险也需要进行管理.

图 4.7 和图 4.8 说明了部分融资债务抵押债券的原理.

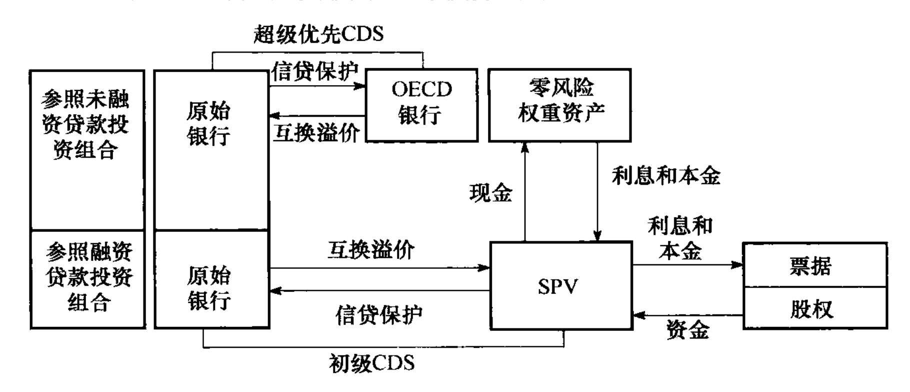


图 4.7 部分用 CDS 和 SPV 融资合成债务抵押债券

资料来源:Investing in Collateralized Debt Obligations, Frank J. Fabozzi, Laurie S. Goodman.

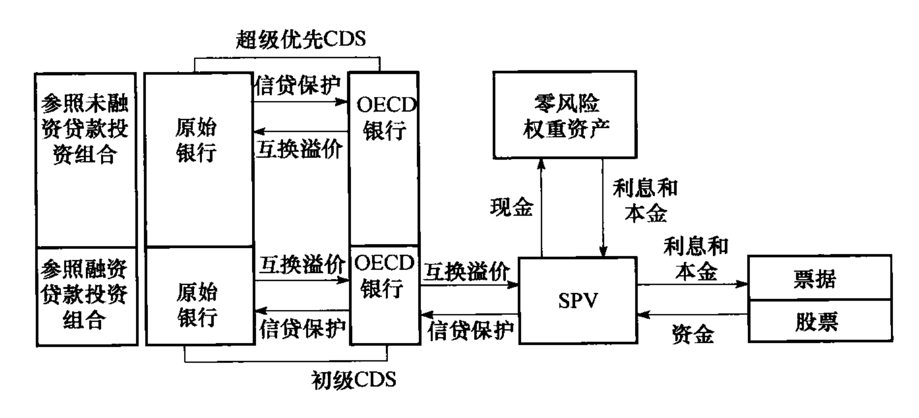


图 4.8 部分用 OECD 银行进行融资合成债务抵押债券

资料来源:Investing in Collateralized Debt Obligations, Frank J. Fabozzi, Laurie S. Goodman.

欧洲银行对待管理资本的方式在各个地区是不同的. 与此相反，美国联邦储蓄则只为美国银行设定了一些特别条款. 在有些情况，我们需要更好地处理管理资本，这时如果能够通过 CDS 将信贷风险转移到另一家 OECD 银行，那么超优先份额就能得到 20%的资本风险权重而资本费用是 8%.

股权部分和低级 CDS 的管理资本规则和全额融资的债务抵押债券的情况相同. 因此，最后一步就是将融资部分和未融资部分的管理资本相加. 如果将这些百分比应用于超优先份额(87%)，那么总管理资本就是 3.4%(4%的超优先 CDS 和 2%的股权部分).

4.3 用 CDS 进行资产负债表管理

银行总在寻求最省钱的融资方式. 因此，很多银行都更偏好于部分融资结构. 图 4.9 比较了全额融资的债务抵押债券和部分融资的债务抵押债券的融资成本.

全额融资的债务抵押债券结构
融资成本34bps

优先票据90%
LIBOR+25 bps

中间票据4%
LIBOR+60 bps

初级票据4%
LIBOR+200 bps

留存股权4%

每份股权全额融资结构的
发行银行需支付17bps

部分融资的债务抵押债券结构
融资成本25bps

超级优先票据87%
LIBOR+14 bps

优先票据3%
LIBOR+25 bps

中间票据4%
LIBOR+60 bps

初级票据4%
LIBOR+200 bps

留存股权2%

每份股权部分融资结构
的发行银行需支付7.26bps

图 4.9 全额与部分合成债务抵押债券的融资成本
资料来源：Picone D(2001).

部分融资的结构能够使银行降低融资成本. 在图 4.9 中，交易成本下降了 9bps(从 34 bps 到 25 bps). 此外，对于每份股权的使用而言，部分融资结构的发行银行只需支付7.26 bps，而全额融资结构则需支付 17 bps.

4.4 资产组合的违约亏损分布

假设图 4.10 中所有的贷款均为 Baa1 级，6 年到期，累计违约率 37%，[18] 回收为 65%. 资产组合的预期亏损为 (1−0.65)×0.37=0.48%. 因此，如果该结构会影响到低级份额的票据持有人，那么预期亏损还应该增长 4 倍. 图 4.10 画出了交易亏损的统计分布.

债务抵押债券结构与利率不匹配无关——例如，一方支付 LIBOR，并且收到 LIBOR. 平均 100 bps的息差补偿了发行银行所承担的标的贷款资产组合的预期亏损风险. 信贷利差在25～200 bps之间，补偿了投资者投资不同票据份额所承担的不同程度的风险. 因此，我们可以在这个结构中减去 LIBOR 而只保留息差，如表 4.9 所示.

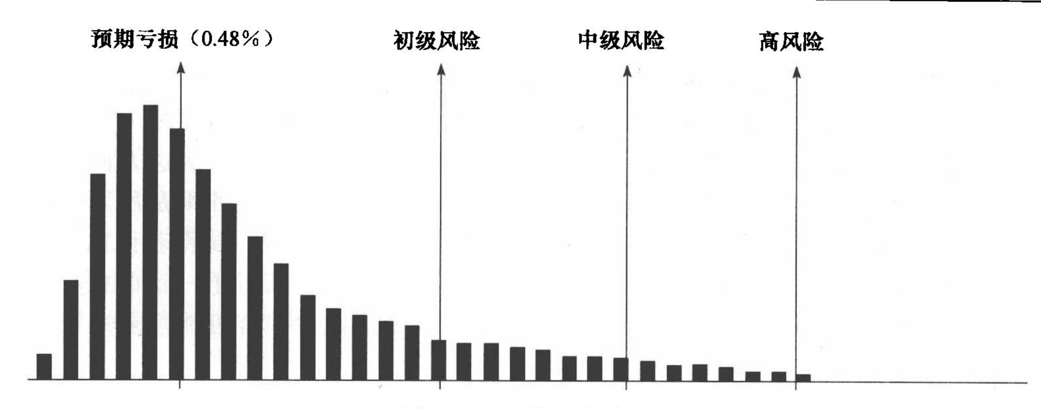


图 4.10 预期亏损分布

资料来源：Picone D(2001).

表 4.9 对冲利率风险中的 CDO 结构

CDO 结构					
资 产	%	息 差	负 债	%	息 差
借贷 1	2%	100 bps	超级优先票据	87%	15 bps
借贷 2	2%	100 bps	优先票据	3%	25 bps
借贷 3	2%	100 bps	中间票据	4%	80 bps
……	…	100 bps	初级票据	4%	200 bps
借贷 50	2%	100 bps	留存股权	2%	股息

资料来源：Picone D(2001)，pg. 15.

总的来说，CDS 的目的是能够模仿浮动利率票据的信贷行为，例如表 4.6 中的几笔贷款. 直到贷款到期，贷款的息差都不变，而这个和 CDS 的固定收益部分相同. 实际上，CDS 的出售者在寻求信贷风险，要么贷款到期要么违约情况发生，否则他们总能够每年获得固定的 X 点的息差收益. 正因为 CDS 的固定收益部分是一个固定的息差，我们可以在资产部分减掉贷款而加上 CDS，如表 4.10 所示.

表 4.10 对冲利率风险中的 CDO 结构与借贷置换

CDO 结构					
资 产	%	息 差	负 债	%	息 差
借贷 1	2%	100 bps	超级优先票据	87%	15 bps
借贷 2	2%	100 bps	优先票据	3%	25 bps
借贷 3	2%	100 bps	中间票据	4%	80 bps
……	…	100 bps	初级票据	4%	200 bps
借贷 50	2%	100 bps	留存股权	2%	股息

资料来源：Picone D(2001)，pg. 15.

对于表 4.10 中的债务抵押债券，银行承担了 50 种合成资产的信贷风险. 为了对冲这部分风险，银行通过 4 种不同信贷风险的票据（例如超优先票据、优先票据、中间票据和低票据）来进行借款. 持有股权则能使其拥有获得分红的机会. 表 4.10 中的某些贷款可能是同一个国家同一行业内的，因此可以假设这几笔贷款受到的信贷风险影响因素是相同的. 因此我们可以将

它们视为一笔贷款，而其票面额为所有贷款票面额的累加. 换句话说，我们可以将债务抵押债券视为一揽子的 CDS，每一个 CDS 承担某一项贷款的信贷风险.

表 4. 11 为一组在资产方票面等值、多样化指数为 30、标的贷款平均评级为 Baa1 的 CDS. 表 4. 11 中的多样化指数表示这 50 笔贷款相当于 30 笔互不相关的贷款. 从这方面来看，这个 CDS 组合是一个做过套期保值的投资组合. 资产是一组合成贷款，而债务则是所发票据的各个有着不同评级的份额. 通过对冲资产表的信贷风险（和利率风险），银行试图获得比投资无风险国债更高的投资回报. 通过对债务抵押债券进行部分融资，银行也获得了一定的杠杆，这个杠杆可能带来巨额回报. 然而，这个对冲交易并非完美，因为预期亏损有可能削弱股权份额上升到优先票据份额.

表 4. 11 CDO 结构与一揽子 CDS

CDO 结构					
资 产	%	息 差	负 债	%	息 差
N 种共同资金		100 bps	超级优先票据	87%	15 bps
等值 CDS		100 bps	优先票据	3%	25 bps
多样化指数：30		100 bps	中间票据	4%	80 bps
平均评级：Baa1		100 bps	初级票据	4%	200 bps
		100 bps	留存股权	2%	股息

资料来源：Picone D(2001)，pg. 16.

我们在接下来的内容中参考 Gibson(2004) 对如何计算债务抵押债券组合违约亏损分布的讨论（也可参考 Vasicek(1987)；Galiani(2003)；Gupton(1997)）. 这些方法都是用了 JP Morgan 建立的 CreditMetrices 框架中的亏损分布计算，而这个框架是根据 Merton 的公司债务定价模型建立的[19]. 为了对合成债务抵押债券各份额进行定价和风险度量，CDS 相关债务组合的违约亏损概率分布是其中的关键. 可以用一个易处理的分析方法计算亏损分布. 假设 CDS 相关债务的违约相关性是由一个共同因素所驱动的. 先以该因素的存在作为条件计算亏损分布，然后再通过积分将条件因素去除，从而得到独立的亏损分布. Gibson(2004) 综合了众多相似的解决这个问题的方法.[20] $i=1, \cdots, N$ 的信贷的相关投资组合中的各部分可用如下参数描述：

A_i　　贷款 i 的票面价值

$q_i(t)$　贷款 i 在 t 时间之前违约的风险中性概率

A_i　　贷款 i 的回收率

名义金额为已知数量，违约风险中性概率可以通过每笔贷款的对应 CDS 息差来进行估计，而回收率假定是一个已知量并且是一个常量. 这种分析使用了从市场息差提取的违约风险中性概率和回收率（与历史违约率或评级转移矩阵不同）. 信贷价值假设取决于参考信用（reference credit）的标准化资产价值 x_i. 当 x_i 下降到某个名义金额阈值 $\bar{x}$ 时，违约情况发生，x_i 的资产价值取决于单一的共同因素 M：

$$x_i = a_i M + \sqrt{1-a_i^2} Z_i \tag{4.1}$$

其中 x_i，M 和 Z_i 都是均值为 0、方差为 1 的随机变量，其分布分别为 F_i，G 和 H_i. 假设 M，Z_1，…，Z_n 为独立分布随机变量，也就是说，$Z_i \sim \Phi(0,1)$. M 表示系统风险因子的标准化回报.

负荷因子 a_i 取值范围是 0～1. 可以将 a_i 看做债务人 i 和市场因素的相关性. 以 a_ia_j 表示信用 i 和信用 j 的资产价值的相关性. 那么违约阈值 $\overline{x}$ 就等于 $F_i^{-1}(p_i(t))$. 条件违约概率 $p_i(t|M)$ 在这一步中也可以记为 Prob $x_i<\overline{x}$：

$$p_i(t|M) = H_i\left(\frac{\overline{x}_i - a_iM}{\sqrt{1-a_i^2}}\right) \tag{4.2}$$

根据式（4.1）或（4.2）得，

$$p_i(t|M) = H_i\left(\frac{F_i^{-1}(p_i(t)) - a_iM}{\sqrt{1-a_i^2}}\right)$$

贷款参考组合（reference portfolio）的亏损分布是一个离散值. 每笔信贷要么不亏损，要么亏损 $A_i(1-R_i)$. 如果每笔信贷对应的 A_i 和 R_i 的值都分别相等，那么可以将下标 i 去掉. 在这样的假设下，亏损分布和违约数目分布就可以相互替换了，只需要将违约数目乘以一个因子 $A(1-R)$ 即可得到亏损数. 如果并非每笔信贷的 A_i 和 R_i 都相等，上述方法仍然适用，只不过计算量会随着组合亏损的离散分布量的增加而增加. 如果离散值的数目变得太大，就必须对分布进行糙化处理（Anderson，Sidenius，Basu(2003)），以便将离散值控制在一个合理的规模内.

基于因子 M 的条件违约数分布可以通过以下递归方式来获得. 令 $p^K(l, t|M)$ 表示在有 K 笔信贷的参考组合中，基于条件 M 恰有 l 笔相关贷款在 t 时刻前发生违约的概率. 假设我们知道其分布：

$$p^K(l,t|M) \qquad l = 0,\cdots,K$$

增加一笔条件违约率为 $q_{K+1}(t|M)$ 的贷款. 这样由 $K+1$ 笔信贷组成的新参考组合的违约分布为：

$$p^{K+1}(l,t|M) \tag{4.3}$$

$$p^{K+1}(0,t|M) = p^K(0,t|M)(1-q_{K+1}(t|M)) + p^K(l-1,t|M)q_{K+1}(t|M)$$

$$l = 1,\cdots,K.$$

$$p^{K+1}(K+1,t|M) = p^K(K,t|M)q_{K+1}(t|M)$$

从当 $K=0$，$p^0(0, t|M)=1$ 的退化违约分布开始，对式（4.3）进行递归运算来求解一个有 N 笔信贷的参考组合的违约分布：

$$p^N(l,t|M) \qquad l = 0,\cdots,N \tag{4.4}$$

在求得了条件违约分布后，非条件违约分布就可以通过如下等式求得：

$$p(l,t) = \int_{-\infty}^{\infty} p^N(l,t|M)g(M)\mathrm{d}M \tag{4.5}$$

其中 g 是因子 M 的概率密度. 该积分可以通过数值积分方法来求解——例如 Simpson 法则.

由于 $p(l, t|M)$ 和 $g(M)$ 函数都非常平滑并且没有跳变情况，用数值积分求解是一个非常直观而快速的方法. 对于每个时间间隔 t，都需要进行 $N+1$ 次数值积分运算，每次积分运算都需要针对不同的 M 值运行多次递归（4）.[21] 这样的计算对于每个可能发生支付行为的时间节点 t 都要进行（例如，5 年期按季付款的债务抵押债券的 20 个支付时间节点）. 到这里，有必要举一个数值范例来说明一下这种违约概率分布的大概形状. 假设 $N=100$，负荷因子 $a_i=\sqrt{0.3}$，风险中性违约概率等于 $1-e^{-0.01t}$（每年违约风险为 1%），$t=1$ 年，共同因子 M 和特征因子（idiosyncratic factor）Z_i 都服从正态分布.

在这样的假设下，表 4.12 给出了条件违约概率分布和非条件违约概率分布的数值示例. 当共同因子增大（也就是向着表的右侧移动）时，发生违约的概率在减小. 而通过等式（4.5）计算出来的非条件违约分布则可以解释为不同条件违约概率分布的加权平均，权重为不同共同因子下的相对违约率.

表 4.12 条件违约概率分布和非条件违约概率分布

违约数目	条件违约概率分布 $p(l, t \mid M)$					非条件违约概率分布 $p(l, t)$
	$M=-2$	$M=-1$	$M=0$	$M=1$	$M=2$	
0	0.001	0.186	0.763	0.971	0.998	0.644
1	0.005	0.316	0.206	0.029	0.002	0.166
2	0.019	0.265	0.028	*	*	0.072
3	0.048	0.147	0.002	*	*	0.039
4	0.087	0.060	*	*	*	0.023
5	0.127	0.020	*	*	*	0.015
6	0.152	0.005	*	*	*	0.010
7	0.154	0.001	*	*	*	0.007
8	0.136	*	*	*	*	0.005
9	0.105	*	*	*	*	0.004
10	0.072	*	*	*	*	0.003
11	0.045	*	*	*	*	0.002
12	0.025	*	*	*	*	0.002
13	0.013	*	*	*	*	0.001
14	0.006	*	*	*	*	0.001
15	0.003	*	*	*	*	0.001
16	0.001	*	*	*	*	0.001
17	*	*	*	*	*	0.001
18	*	*	*	*	*	*
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	*	*	*	*	*	*

注：* = 小于 0.001.

资料来源：Gibson M(2004)，pg. 7.

4.5 债务抵押债券的股权份额

债务抵押债券的股权份额可视为混合证券. 其特征为一份付息债券、一份公司股权、一份针对抵押资产的看涨期权和一份管理基金. 作为一项付息债券，债务抵押债券股权是以面值或接近面值的价值发行的并且附有一个最终到期日. 如同可转换债券，其付息多少没有在合同中明确指出，尽管在发行时就有一个预期付息的分布范围. 和看涨期权相类似，债务抵押债券的股权价值随着标的资产的价格和波动性的增加而增加. 与任何动态管理的投资相同，资产经理的表现是影响债务抵押债券股权份额业绩的一个关键因素.

4.5.1 债务抵押债券股权份额的业绩

债务抵押债券的股权份额相当于一项标的资产的杠杆投资，属于资产经理的资产管理技

巧范畴. 杠杆是通过发行投资和次投资级的债务来实现的，这些债务以结构性资产抵押证券形式发行. 很明显，信贷损失是影响债务抵押债券股权份额业绩的重要因素. 它能以两种方式影响投资者. 首先，由于债务违约（现金流型债务抵押债券）或实际价值下降（市场价值型债务抵押债券）造成抵押资产缩水，那么标的资产同样减少，而根据标的资产大小收到的利息也会相应减少. 其次，当抵押资产的票面价值下降到了由某些评级机构评定的触发值（由过量担保测试和利息覆盖测试得到）时，那些原来向股权持有者支付的额外利息就会向优先票据持有者支付，从而造成债务抵押债券的杠杆缩小. 只有当抵押资产面值与债务比回到触发值之上时，额外利息才会重新向股权持有者发放. 交易造成的利息收入和利息费用比的下降也会造成股权份额收益分配的改变. 在进行一个债务抵押债券交易时，剩余的分布于股权份额的利息就等于超额利差，它一般在 2.5%～3%之间，从而给股权投资者带来 25%～30%的收益回报.

4.5.2 债务抵押债券的内嵌期权

根据抵押资产类型和交易时间的不同，债务抵押债券股权份额的内嵌看涨期权可能就变得很有价值. 图 4.11 展示了 Goldman Sachs Single B 级债券指数针对 LIBOR 的历史息差和债务抵押债券的预期融资成本——该图表明了随着资产收入和债务成本之间的差值的增大，债务抵押债券股权份额的投资动机也随着增大. 行使看涨期权带来的价值上涨很大程度上依赖于抵押资产的类型，这样使得将债券抵押债券（CBO）和贷款抵押债券（CLO）区分对待变得势在必行.

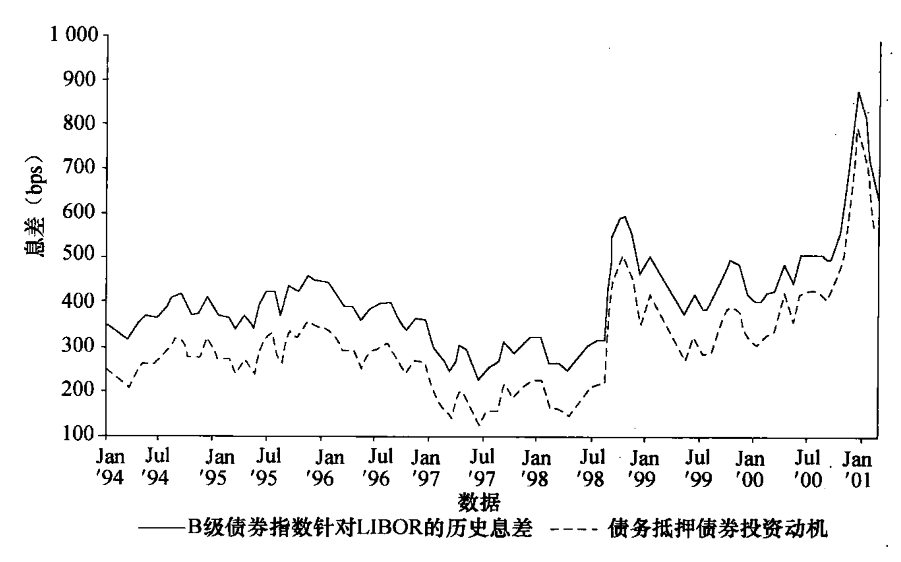


图 4.11 Goldman Sachs B 级债券指数针对 LIBOR 的历史息差和债务抵押债券的预期融资成本
资料来源：Picone D(2001)，22.

对于那些抵押资产是低价收购的债券（当利率很高时）而融资结构是便宜的期限型票据（现行的低利率）的债务抵押债券，它们能够提供最大可能性的资产增值. 例如，杠杆贷款

这类浮动利率抵押资产可以方便地进行再融资. 标的资产的债务方能够提前偿还大部分贷款而以一个更低的息差来获得再融资. 因此，贷款抵押债券经理就很难产生巨大的资产增值. 随着剩余部分的预期回报的下降，股权持有者很可能在偿还阶段行使他们的期权. 目的在于抓住债券抵押债券的潜在回报的机会或者能够使得贷款抵押债券的糟糕的信贷环境所造成的影响降到最小.

4.5.3 股权份额的价格

一旦出现本金赎回的情况，债务抵押债券股权份额的价格就会有一个自然的下降趋势. 图 4.12 展示了一个债券抵押债券交易的现金流数据分布. 该分布按季度结算. 下篇文章详细分析了同一个债券抵押债券. 根据一个固定不变的年违约率为 3%的假设，我们创建了一个股权分布范例. 股权分布直到第 17 季度才收到利息，而余下的本金在其后的 4 个季度收到，在生成这个支付过程的时间序列中，我们假定没有看涨期权的交易发生. 而实际上，当杠杆下降时，股权投资者很有可能来行使一个表现良好的资产的看涨期权，这一般发生在第 6 年和第 8 年之间.

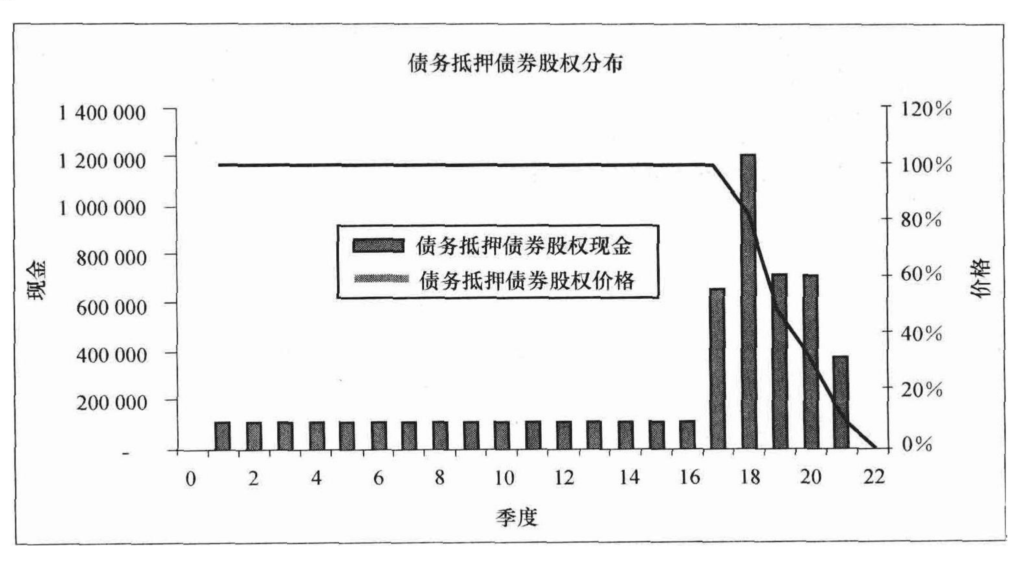


图 4.12 债务抵押债券股权份额的价格与现金分布

资料来源：Picone D(2001)，25.

我们期望债务抵押债券股权的价格随着时间一直在改变. 其他因素，诸如抵押资产组合的价值变化、看涨期权的价值、票据所附的浮动利率的变化都会影响到股权价格的走势. 由于债务抵押债券股权相当于一个标的为抵押资产的看涨期权，我们可以预计债务抵押债券股权份额的价格随着到期日的临近而下跌. 浮动利率的改变影响了抵押资产所带来的收入和发放票据的融资成本之间的息差. 浮动利率的上涨挤压了交易的利差，这个利差可以用来弥补交易的亏损. 我们也可以预计杠杆同样会影响股权份额的收益指数. 图 4.13 展示了两个有不同亏损率的债务抵押债券股权：杠杆更高的股权份额收益在亏损率小于 6%之前都更高，而在亏损率大于 6%之后反而亏损得更严重. 此外，高杠杆的交易可能带来更严格的过量担保

测试水平，从而会使该结构更快地达到触发值．我们使用之前相同的债务抵押债券结构来产生这个回报曲线．

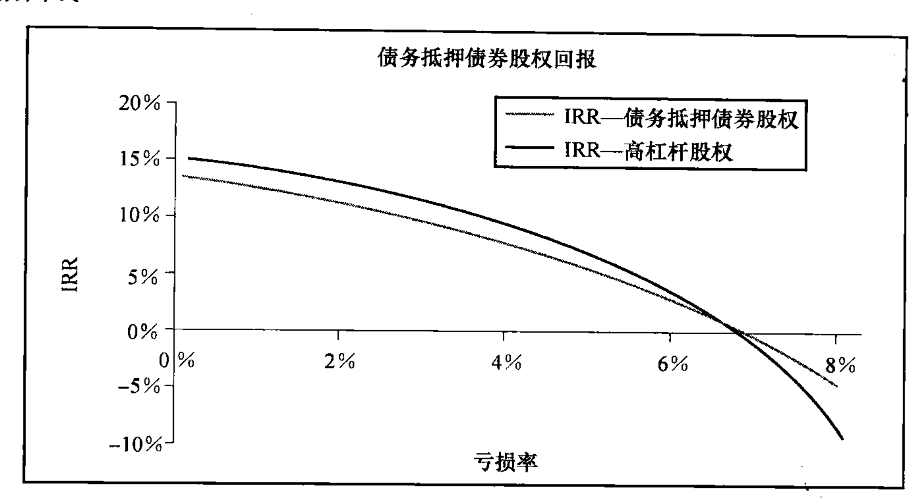


图 4.13　两个有不同亏损率的债务抵押债券股权投资者的回报

资料来源：Picone D(2001)，25.

4.5.4　使用穆迪二项展开技术来构建合成债务抵押债券

穆迪（Moody）使用二项展开技术（BET）来计算抵押资产承担的信贷风险．通过多样化指数，BET 减少抵押资产违约概率之间的相关性，使之成为违约概率不相关的一组同质资产．多样化指数 D 能提供互不相关的一组债券或贷款，但能够模仿原来抵押资产组合的表现行为．比如，在到期时 D 中的债券可能违约或不违约——也就是说，一共只有两种可能，违约或者不违约．此外，一种债券的违约概率和其他债券的违约不相关．在这样的假设下，D 中的 N 种债券违约的概率可以用二项分布 $P \sim Binomial(D,\ N,\ p)$ 来计算：

$$P_N = \binom{D}{N} p^N (1-p)^{D-N} \tag{4.6}$$

其中，p 表示经过适当的因子调整后组合中资产的平均违约率．

一旦抵押资产的风险确定后，将其与为所有债务抵押债券份额提供的保证某个结构能够达到一定评级水准的信贷担保进行比较．当违约行为发生时，亏损首先波及低级票据，随后是中间票据，最后是优先票据．计算结果可以通过对整个交易期间可能产生的违约行为进行模拟仿真来获得．从最初的状态下不产生违约开始，每个同质债券（homogeneous bond）都通过二叉树的分支来模拟仿真直至到期日，每个分支违约概率为 p，不违约概率为 $1-p$．然后再计算债务抵押债券结构可能承担的亏损，并将其与表 4.13 穆迪理想累积期望亏损比较．例如，对于一组平均到期时间为 5 年的抵押资产，其 Aaa 级优先票据的最大累积亏损不能超过 0.002%．

表 4.13 穆迪理想累积期望亏损

	年									
	1	2	3	4	5	6	7	8	9	10
Aaa	0.000%	0.000%	0.000%	0.001%	0.002%	0.002%	0.003%	0.004%	0.005%	0.006%
Aa1	0.000%	0.002%	0.006%	0.012%	0.017%	0.023%	0.030%	0.037%	0.045%	0.055%
Aa2	0.001%	0.004%	0.014%	0.026%	0.037%	0.049%	0.061%	0.074%	0.090%	0.110%
Aa3	0.002%	0.010%	0.032%	0.056%	0.078%	0.101%	0.125%	0.150%	0.180%	0.220%
A1	0.003%	0.020%	0.064%	0.104%	0.144%	0.182%	0.223%	0.264%	0.315%	0.385%
A2	0.006%	0.039%	0.122%	0.190%	0.257%	0.321%	0.391%	0.456%	0.540%	0.660%
A3	0.021%	0.083%	0.198%	0.297%	0.402%	0.501%	0.611%	0.715%	0.836%	0.990%
Baa1	0.050%	0.154%	0.308%	0.457%	0.605%	0.754%	0.919%	1.085%	1.249%	1.430%
Baa2	0.094%	0.259%	0.457%	0.660%	0.869%	1.084%	1.326%	1.568%	1.782%	1.980%
Baa3	0.231%	0.578%	0.941%	1.309%	1.678%	2.035%	2.382%	2.734%	3.064%	3.355%
Ba1	0.488%	1.111%	1.722%	2.310%	2.904%	3.438%	3.883%	4.340%	4.780%	5.170%
Ba2	0.858%	1.909%	2.849%	3.740%	4.626%	5.374%	5.885%	6.413%	6.958%	7.425%
Ba3	1.546%	3.030%	4.329%	5.385%	6.523%	7.419%	8.041%	8.641%	9.191%	9.713%
B1	2.574%	4.609%	6.369%	7.618%	8.866%	9.840%	10.522%	11.127%	11.682%	12.210%
B2	3.938%	6.419%	8.553%	9.972%	11.391%	12.458%	13.206%	13.833%	14.421%	14.960%
B3	6.391%	9.136%	11.567%	13.222%	14.878%	16.060%	17.050%	17.909%	18.579%	19.195%
Caa	14.300%	17.875%	21.450%	24.134%	26.813%	28.600%	30.388%	32.174%	33.963%	35.750%

D 个同质资产中某一项的违约亏损通过计算违约债券相关的现金流现值来获得，并用回收率来对其进行调整.

该事件发生的概率是：

$$EL_1 = P_1 * L_1 \tag{4.7}$$

预期亏损通过对每种资产在各种情况下的违约损失累加求和来获得，$N=0$，1，2，…，D，

$$EL = \sum_{N=0}^{D} P_N * L_N \tag{4.8}$$

预期外损失为：

$$UL = \sum_{N=0}^{D} P_N * (L_N - EL)^2 \tag{4.9}$$

因此，为了使用 BET，就必须计算如下抵押资产的变量：违约概率、亏损和多样化指数.

有时，抵押资产组合可能由两个（或更多）高度不相关的资产组成，它们的平均特性不相同. 在这种情况下，穆迪用了与 BET 有微小区别的方法为其建模，这种方法称为双 BET. 使用双 BET 时，我们将它们视为两类相互独立的资产组合. A 类资产中 a 个资产的违约概率和 B 类资产中 b 种资产的违约概率为两个相互独立的变量，其分布可以写成 $P \sim Binomial$（D_A，D_B，N_A，N_B，p_A，p_B）：

$$P_{a+b} = \binom{D_a}{a} p_A^a (1-p_A)^{D_a-a} \binom{D_b}{b} p_B^b (1-p_B)^{D_b-b} \tag{4.10}$$

$a+b$ 个违约所带来的亏损可以通过 $a+b$ 个债券违约带来的现金流现值与总现金流的现值来计算.

整个资产类的预期亏损可以通过对所有情况下（$a+b=0$，1，2，3，…，N_A+N_B）发生

的预期亏损进行累加来获得.

$$EL = \sum_{i=0}^{D_a} \sum_{j=0}^{D_b} P_{ij} L_{ij} \tag{4.11}$$

而预期外亏损为：

$$UL = \sum_{i=0}^{D_A} \sum_{j=0}^{D_B} P_{ij} (L_{ij} - EL)^2 \tag{4.12}$$

违约概率

违约概率通过抵押资产的评级来计算. 当公开评级无法获得时，穆迪会得出一个影子（shadow）评级. 然后将标的资产的到期日考虑进去进行调整来得到一个累积违约概率. 抵押资产的累积概率通过对所有资产的累积违约概率进行加权求平均获得，而每个权重则为资产的面值：

$$CDP = \frac{\sum_{N=0}^{M} CP_N \cdot A_N}{\sum_{N=0}^{M} A_N} \tag{4.13}$$

其中，CP_N 为债券 N 的累积违约概率，A_N 为债券 N 的面值，M_N 为总资产数 .

亏损严重程度

亏损的严重程序取决于假定的回收率值和回收时间. 穆迪假定回收率不受资产评级影响，但依赖于债务的优先级和安全性. 穆迪还假设基本情况下的回收率至少为市值的 30%或者面值的 25%. 对于新兴市场，回收率降到最小为市值的 20%或面值的 15%. 不同贷款份额的历史回收率在表 4.14 中给出.

表 4.14 不同贷款份额的历史回收率

贷款/债券	回收率
高级抵押贷款	70%
高级无抵押贷款	—
高级抵押债券	52%
高级无抵押债券	49%
次级债券	33%

资料来源：Picone(2001) .

多样化指数

穆迪通过多样化指数解决了估计违约相关性的问题，度量了在不相关的资产类中，可能经历原始资产组合中的违约程度的资产数量. 由于在多样性较差的资产组合中违约相关性较高，较低的多样化指数意味着风险更高的投资组合. 用行业分类来计算多样化指数. 行业聚集性通过用债券的面值作为权重来计算. 在行业聚集性计算出来以后，使用表 4.15 中的多样化指数那一栏来计算多样化指数.

表 4.15 多样化指数

不同行业的债券发行商数量	多样化指数	拉丁美洲的多样化指数	不同行业的债券发行商数量	多样化指数	拉丁美洲的多样化指数
1.00	1.00	1.00	4.00	2.30	1.65
1.50	1.20	1.10	4.50	2.50	1.75
2.00	1.50	1.25	5.00	2.70	1.85
2.50	1.80	1.40	5.50	2.80	1.90
3.00	2.00	1.50	6.00	3.00	2.00
3.50	2.20	1.60			

资料来源：Picone(2001) .

穆迪同样将所有地区新兴市场发行的债券的多样化指数进行区分. 从表 4.12 可知，由四个新兴市场债券组成的组合的多样性指数为 2，而同样数目的美国高收益债券组合的多样性指数为 3. 为了求拉丁美洲的多样化指数，则要对其进行一下调整：

$$LADS = 1 + (DS - 1) \times 0.5$$

4.5.5 债务抵押债券各层份额的相关性风险

正如 Gibson 所说，在整个债务抵押债券周期内，债务抵押债券结构的定价反映了投资者对参考组合内资产违约的相关性的预期. 市场报价反映了市场对投资组合结构内不同位置的相关性的看法，从而反映了市场对组合亏损分布形状的看法. 由于这类相关性是无法观测的，因此债务抵押债券份额暴露在相关性风险下. 在给出了数目为 $N=100$ 笔信贷的参考组合信贷之间的相关性后，图 4.14 说明了三种假想的典型债务抵押债券结构的市场价格有何不同（假设资产组合中信贷的相关性为 0.3，使用表 4.16 中的面值差额，对资产中各份额按照信贷相关性从 0 到 0.9 进行重新估价）.

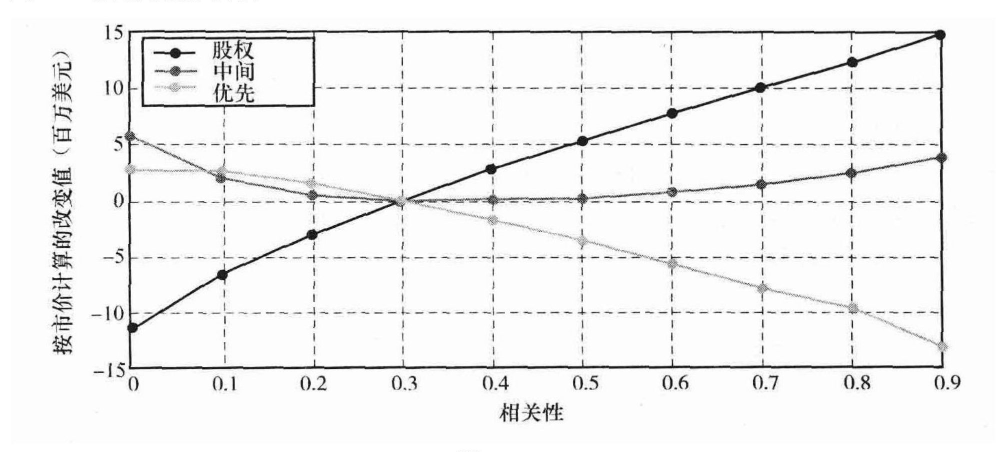


图 4.14

资料来源：Gibson M(2004)，19.

某一结构的违约相关性越高，那么就越有可能会使损失侵蚀掉股权和中间份额并且对优先层票据造成损失.[23] 因此，当相关性增加时，优先份额的价值就会下降. 反过来，更高的相关性也使得更有可能出现几乎不发生违约这种极端情况，从而导致股权份额的价值随着相关性的增加而上升. 和在很多违约发生时蒙受的损失相比，股权份额投资者在几乎不发生违约这种情形下获得的回报要更多（这些投资者只受最初几笔违约的风险影响）.[24] 中间票据则同样受到两个因素的影响，因此可以将两种因素相互抵消，从而使得中间份额的票据价值对于相关性的敏感度要小得多，正如图 4.14 所示. 相关性影响对信贷市场近期的革新尤其严

表 4.16

发行份额	连接点（百分比）	名义总额（百万美元）	面值差额
股权	0～3	30	1507
中间份额	3～10	70	315
优先份额	10～100	900	7
整个投资组合	0～100	1 000	60

资料来源：Picone M(2004)，pg. 11.

重，包括那些单份额的债务抵押债券和第一违约偿还制的掉期合约（在第 5 章中进行讨论）.

为了能够求得 I 和 j 这两个债务人的违约相关性，我们使用两个随机变量的相关性公式：

$$\mathrm{Corr}(x_i,x_j)=\frac{\mathrm{Cov}(x_i,x_j)}{\sqrt{\mathrm{var}(x_i)\mathrm{var}(x_j)}}=\frac{E(x_ix_j)-E(x_i)E(x_j)}{\sqrt{p_i(1-p_i)}\sqrt{p_j(1-p_j)}}=\frac{p_{ij}-p_jp_j}{\sqrt{p_i(1-p_i)}\sqrt{p_j(1-p_j)}}$$

其中 x_i 为伯努利型随机变量，均值为 p_i，标准差为 $\sqrt{p_i(1-p_i)}$. 两个债务人的联合违约概率 p_{ij} 可以通过资产回报的联合分布来获得. 而 p_i 的值可以通过公式（4.2）来求得.

内含相关性有两种度量方法：复合相关性和基本相关性. 复合相关性法对各份额都区别处理. 首先选择一个定价模型，然后将每个份额的一个（保持不变的）相关性作为模型的输入来求得定价.[25]通过迭代处理，复合相关性可以作为输入相关数来求得，这个相关性造成了和市场报价相等的利差.[26]这种方法的一个缺陷就是无法求得唯一解，因为复合相关性既是一个对该层份额上限连接点的函数，也是对下限连接点的函数.[27]

基本相关性法由 JP Morgan 提出. 和复合相关性相反，利用自举剥离过程，基本相关性法同时考虑多个份额的值.[28]假设有一个份额化的信贷产品的上限连接点为 $K_j(j=1, \cdots, n)$，令 ρ_{K_j} 为某个上限连接点为 K_j 的份额的基本相关性. 第一个股权份额的预期亏损 $EL_{\rho_{K_1}}(0, K_1)$ 已经在市场上交易，而非股权份额的预计亏损 $EL(K_1, K_2)$，$EL(K_2, K_3)$，……也一样在市场上交易. 其他股权份额的预期亏损可以用自举法来计算：

$$EL_{\rho_{K_2}}(0,K_2)=EL_{\rho_{K_1}}(0,K_1)+EL(K_1,K_2)$$

$$EL_{\rho_{K_2}}(0,K_3)=EL_{\rho_{K_1}}(0,K_2)+EL(K_2,K_3)$$

依此类推. 按照 McGinty 和 Ahluwalia(2004）的说法，在标准化的大资产组模型下，基本相关性的定义是，能够使得份额价值与利差报价相等的一系列股权份额的相关性输入值.[29]关于基本相关性的讨论可参见 Willemann(2004).

关于相关性模型检验和 $\mathrm{CreditRisk}_+$（CR_+）模型的内含相关性校准来重新生成 iTraxx 欧洲份额报价的工作，可参见 Lehnert、Altrock、Rachev、Trück 和 Wilch(2005). 他们发现，与市场上优先份额的亏损相比，如果使用初始参数，那么 CR_+ 模型产生的厚尾现象要小很多.

于是，他们总结认为为了能重新产生出这些市场报价，需要对 CR_+ 的参数进行一些调整，以便能够模仿出这样的厚尾亏损分布.

4.6 债务抵押债券份额定价

由于债务抵押债券的相关性结构会影响定价结果，可以使用 copula 函数框架（如高斯 copulae 和学生 t copulae）来对其进行定价. 我们按照 Galiani(2003）中的讨论（也可以参照 Gibson(2004）以及 Gregory 和 Laurent(2002）中对于债务抵押债券定价的讨论).

我们来定义一下定价中的符号表示：

- n 表示抵押资产组合中的资产数.
- N_i 表示第 i 项参考债务的名义金额.
- R_i 表示第 i 项参考债务的回收率.
- $T=t_n$ 表示合约的法定到期时刻，从当前的 $t_0=0$ 开始以年计.

- τ_i 是第 i 个债务人的违约停止时刻，也就是 Cox 过程 n 中第一个跳变发生时刻，跳变密度为 $\lambda_i(t)$.
- $D(0, t)$ 表示无风险贴现利率（假定为恒定不变）.
- s 表示债务抵押债券份额的公平价格，以每年点数计算，这个点数表示名义现存份额的比例.

以 $L_i=(1-R_i)N_i$ 和 $Q_i(t)=1_{\tau_i<t}$ 分别表示第 i 个债务人在 t 时刻违约造成的亏损和违约发生标识. 违约发生标识在 $\tau_i<t$ 时为 1，其余时刻为 0. 定义 $L(t)$ 为抵押资产组合的累积亏损：

$$L(t) = \sum_{i=1}^{n} L_i Q_i(t) = \sum_{i=1}^{n} L_i 1_{\{\tau_i<t\}} \tag{4.14}$$

票据持有人之间的亏损分布随着所持份额的优先级不同而存在差异. 当抵押资产的亏损超过某一下限而小于某一上限时，某一份额的违约部分只能间断地收到本利支付，这个下限和上限分别为 A 和 B. A 和 B 分别称为该份额的**连接点**（atlachment point）和**脱离点**（de-attachment point），在这两个点间，可能造成对该份额的损失在 A 和 B 之间. 例如，在一个三层份额的债务抵押债券结构中，如果 A 和 B 之间的份额是中间票据，那么低于 A 的损失只会发生在股权份额而高于 B 的损失发生在优先份额中. 因此，某个份额的优先级通过两个连接点 A 和 B 的相对位置来确定. 如果 $A=0$，那么该份额就是股权份额；如果 $A>0, B<\sum_{i=1}^{n} N_i$，那么该份额为中间份额；而如果 $B=\sum_{i=1}^{n} N_i$，那么该份额为优先份额.

因此，如果 $L(t)<A$，那么对于一个给定份额的累积亏损 $L^{A,B}(t)$ 就为 0；如果$A\leqslant L(t)<B$，则 $L^{A,B}(t)$ 等于 $L(t)-A$；如果 $L(t)\geqslant B$，则 $L^{A,B}(t)$ 为 $B-A$. 将其进行整理得到：

$$L^{A,B}(t) = (L(t)-A)1_{[A,B]}L(t) + (B-A)1_{\{B,\sum_{i=1}^{n}N_i\}}L(t) \tag{4.15}$$

4.7 定价公式

和一揽子违约掉期的定价分析方法类似，债务抵押债券份额的合理价格是在违约金额（DL）同溢价收入（Premium Leg，PL）相等的条件下推得的.

使用 Laurent 和 Gregory(2002) 中的表示方法，我们用 DL 来表示违约行为若不发生所产生的现金流从违约发生时刻进行贴现的期望值：

$$DL = E^*\left[\int_0^T D(0,t)\mathrm{d}L^{A,B}(t)\right] \tag{4.16}$$

而 PL 用来表示溢价收入=0 现值的期望，以每个支付日的资本量作为权重，每年支付$1/\alpha$次[30]：

$$PL = E^*\left[\alpha\sum_{i=1}^{n} s_{A,B}D(0,t_i)\min\{\max[B-L(t_i),0],B-A\}\right] \tag{4.17}$$

在等式（4.17）中，我们应该注意到，在抵押资产组合中没有发生违约的情况下（或者违约数存在一个上限从而使得累积亏损量小于 A），经过贴现的溢价收入是以份额的总的名义金额进行加权的. 而累积亏损在 A 和 B 之间时，参考名义金额的总量减少了，直到减少至 0，而此时累积亏损超过了上限连接点 B.

因此债务抵押债券份额的公平价格可以定义为一个利差 $s_{A,B}^*$，使得

$$s_{A,B}^{*} = PL\ (s_{A,B}^{*}) - DL\,(s_{A,B}^{*}) = 0$$

于是可以得到：

$$s_{A,B}^{*} = \frac{E^{*}\left[\int_{0}^{T} D(0,t)\,\mathrm{d}L^{A,B}(t)\right]}{E^{*}\left[\alpha\sum_{i=1}^{n} D(0,t_i)\min\{\max[B-L(t_i),0],B-A\}\right]}. \qquad (4.18)$$

尽管以上公式看起来简洁明了，但是为了计算累积亏损分布从而能进一步计算利差 s^* 所涉及的计算还是相当繁复的，这些将在下一节进行讨论.

4.8 仿真算法

在分别给出了债务抵押债券份额的损失上、下限值之后，我们可以用蒙特卡罗仿真来估计债务抵押债券抵押资产的亏损分布情况并以此来计算该份额的利差 $s_{A,B}^{*}$. 定价过程的第一步和定价一揽子违约掉期的情况非常相似. 此外，由于该问题的高维数（抵押资产组合有超过 50 个的参考实体），最好能够使用一定的方差消减技术来减少收敛所需的仿真路径数. 该过程可以总结如下：

1. 对于每个参考实体，对时间 $t=T_1$，T_2，…，T_M 计算内涵违约强度 $\lambda_n(t)$，其中 t 为市场现有信贷违约掉期的到期日期的集合；从而校准违约时间分布函数的参数.

2. 校准为建立组合中债务人违约相关性模型选择的 copula 函数的参数（参考 2.4 节）.

3. 对于每次仿真 k，重复以下步骤：

a) 使用 5.9.2 节中所介绍的算法产生一个 n 维的相关的 0－1 分布随机向量.

b) 对于每个债务人，使用 5.9.3 节中的算法将相应的 0－1 分布随机变量转换成违约时间.

c) 将 n 维违约时间向量 τ^k 从小到大进行排序，并从中选出一组违约时间向量 $\Gamma^k=\{\tau_1^k$，τ_2^k，…，$\tau_n^k\}$，使得对于任意 $j\in\{1, 2, \cdots, n\}$ 都满足 $T_j^k\leqslant T$.

d) 根据如下步骤计算离散的违约金额：

ⅰ. 使用公式（4.1），对于某个向量 Γ^k，通过计算 $L^k(t)$ 来得到抵押资产组合的累积亏损.

ⅱ. 若 $L^k(t)<A$，违约支付量设为 0.

ⅲ. 若 $A\leqslant L^k(t)<B$，从中选择一个违约触发时刻 $\tau_a^k=\inf\{t>0\,|\,L(t)\ \geqslant A\}$，而对于每一个违约时刻大于或等于 τ_a^k 的违约债务人 $\ell\in\{1, 2, \cdots, L\}$，计算他们的违约金额折现值 DP_w^k，他们的违约时刻集合为 $\Gamma_{A,B}^k=\{\tau_a^k$，$\tau_{a+1}^k$，…，$\tau_L^k\}$. 对于 $w\in\{a, a+1, \cdots, L\}$，利用 $DP_w^k=D(0,\ \tau_w^k)L_w=D\ (0,\ \tau_w^k)\ (1-R_w)N_w$ 这个公式得到违约金额折现值，并将所有违约债务人的违约金额折现值累加，即 $DP^k=\sum_{w=a}^{L}DP_w^k$.

若 $L^k(T)\geqslant B$，那么使用先前的方法得到下限违约触发时刻 τ_a^k，并计算违约上限触发时刻 $\tau_b^k=\inf\{t>0\,|\,L(t)\geqslant B\}$. 对于每个违约债务人 $\ell=\{1, 2, \cdots, L\}$，他们的违约时刻集合大于等于 τ_a^k 而小于 τ_b^k，计算相应的违约金额贴现值，即给定他们的违约时刻集合为 $\Gamma_{A,B}^K=\{\tau_a^k$，$\tau_{a+1}^k$，…，$\tau_b^k\}$，计算 $DP_w^k=D(0,\ \tau_w^k)L_w=D(0,\ \tau_w^k)(1-R_w)N_w$，其中 $w\in\{a, a+1, \cdots, b\}$，最后累加所有违

约债务人的违约金额贴现值，即 $DP^k = \sum_{w=a}^{b} DP^k_w$.

e）根据如下步骤计算溢价收入：

对于每个向量 τ^k 中的每个溢价日 $\{t_1, t_2, \cdots, t_n\}$，计算相应的累积亏损 $L^k(t_i)$，$i \in \{1, 2, \cdots, n\}$，然后计算溢价收入

$$PL^k = \alpha \sum_{i=1}^{n} D(0,t_i)\min\{\max[B - L^k(t_i),0],B - A\}$$

4. 计算 DP^k 和 PL^k 的算术平均数并利用等式（4.5）来确定公平利差 $S^*_{A,B}$.

4.9 在 Matlab 中的债务抵押债券定价[31]

如下的 Matlab 函数是用来进行债务抵押债券份额定价的. CDO_tranch. m 为各份额定价并通过调用 gaussian_time. m 来生成违约时间向量，通过 cash_flow. m 来产生贴现的现金流.

CDO_tranche.m

```
function [price_eq,price_mezz,price_sen]=CDO_tranche(ref_ent,T,k)

% this function computes the fair price of a synthetic CDO tranched in three
% portions, 0-3%, 3-14% and 14%-100%. Spread for each reference entity is
% fixed @ 150 bps. We let recoveries and pairwise correlation changing in
% order to produce a chart of CDO breakeven spread vs. correlation &
% recovery.

% Inputs:

% ref_ent:  # of reference entities in the pool
% T:        # maturity of the deal
% k:        # of simulation


tic
%inizialize a vector of zeros for each of the three tranches
CDO_P1=zeros(5,5);
CDO_P2=zeros(5,5);
CDO_P3=zeros(5,5);
%inizialize a vector of zeros for different recoveries
Recovery =zeros(ref_ent,5);
hazard=zeros(5,1);
% set the obligors' spread to 150bps p.a.
spread =150/10000;

% since the CDS term structure is flat, the relationship hazard
% rate=spread/(1-recovery) holds. Therefore, for each recovery we calculate
% the corresponding hazard rate.
for rec_cycle=1:5
    Recovery(:,rec_cycle)=(.2*rec_cycle)-.2; % recovery ranges from
                                             % 0% to 80%
    hazard(rec_cycle)=spread/(1-Recovery(1,rec_cycle));
end

ZC=0.05; % constant interest rate

Amount=zeros(ref_ent,1); %vector of notional amount for each credit
Amount(:)=100; % each credit has a notional amount of 100 units
```

```
C=zeros(3,1); % we fix three attachment points: 0%,3%,14%
D=zeros(3,1); % we fix three detachment points: 3%,14%,100%
C(1)=(0/100)*sum(Amount);
D(1)=(3/100)*sum(Amount);
C(2)=(3/100)*sum(Amount);
D(2)=(14/100)*sum(Amount);
C(3)=(14/100)*sum(Amount);
D(3)=(100/100)*sum(Amount);

time=zeros(ref_ent,1);
index=zeros(ref_ent,1);

R=[0:.2:0.8]; %constant pairwise correlation ranges from 0% to 80%
% start the correlation loop
for R_cycle=1:5
    for xx=1:ref_ent
       for yy=1:xx
          if xx==yy
             corr(xx,yy)=1;
          else
             corr(xx,yy)=R(R_cycle); %populate the correlation matrix
             corr(yy,xx)=R(R_cycle);
            end
        end
     end
     def_t=gaussian_time(corr,k,ref_ent); % generate pseudodefault times
                                          % with gaussian copula and
                                          % constant hazard rate


     S_fees=zeros(5,3);      % dummy variable for memorizing the simulated
                             % payment leg
     S_default=zeros(5,3);   % dummy variable for memorizing the simulated
                             % default leg
     M_fees=zeros(5,3);      % variable which memorize the payment leg for
                             % each loop of recovery&corr
     M_default=zeros(5,3);   % variable which memorize the payment leg for
                             % each loop of recovery&corr

     for n=1:k % start the simulation loop
        for rec_cycle=1:5 %start the recovery loop
           [time,index]=sort(def_t(n,:)); % sort the pseudo vector of
                                          % default times
           tau=[time./hazard(rec_cycle);index]; % generate the vector
                                                % of default time by
                                                % dividing by the
                                                % corresponding hazard
                                                % rate
           for u=1:3 % start the loop for each of the three tranches
               recovery=0;
               fees=0;
               % calculate the premium and default leg
               [default,fees]=cash_flow(T,tau,Recovery(1,rec_cycle),ZC,
                 Amount(1),C(u),D(u));
               S_fees(rec_cycle,u)=S_fees(rec_cycle,u)+fees;
               S_default(rec_cycle,u)=S_default(rec_cycle,u)+default;
           end
        end

     end
     for u=1:3
```

```
        for rec_cycle=1:5
            M_fees(rec_cycle,u)=S_fees(rec_cycle,u)/k; % average DV01
            M_default(rec_cycle,u)=S_default(rec_cycle,u)/k; % average
                                                        % default leg
        end
    end

    for rec_cycle=1:5
        CDO_P1(R_cycle,rec_cycle)=(M_default(rec_cycle,1)/
          (M_fees(rec_cycle,1)))*10000; % B\E spread for the 0-3% tranche
        CDO_P2(R_cycle,rec_cycle)=(M_default(rec_cycle,2)/
          (M_fees(rec_cycle,2)))*10000; % B\E spread for the 3-14% tranche
        CDO_P3(R_cycle,rec_cycle)=(M_default(rec_cycle,3)/
          (M_fees(rec_cycle,3)))*10000; % B\E spread for the14-100%tranche
    end
end

price_eq=CDO_P1;
price_mezz=CDO_P2;
price_sen=CDO_P3;

figure(1)
surf((0:.2:.8),R,price_eq);
title('Equity Tranche (0%-3%)')
xlabel('Recovery');
ylabel('Correlation');
zlabel('Tranche spread (bps per annum)');

figure(2)
surf((0:.2:.8),R,price_mezz);
title('Mezzanine Tranche (3%-14%)')
xlabel('Recovery');
ylabel('Correlation');
zlabel('Tranche spread (bps per annum)');

figure(3)
surf((0:.2:.8),R,price_sen);
title('Senior Tranche (14%-100%)')
xlabel('Recovery');
ylabel('Correlation');
zlabel('Tranche spread (bps per annum)');

toc
```

cash_flow.m

```
function [PV_def, PV_premium]=cash_flow(expiry,def_time,rec,zc_rate,
  capital,C,D)
% This function computes the value of the premium (DV01, spread still to be
% determined) and default legs of a CDO. Inputs are:

% expiry: CDO maturity
% def_time: simulated default time
% rec: recovery
% zc_rate: constant ZC rate
% capital: nominal amount of each reference entity
% C: attachment point
% D: detachment point
```

```
PV_def=0;
PV_premium=0;
num=size(def_time,2);
loss=zeros(num,1); % loss give default for each credit
tot_loss=0; % cumulative portfolio loss
periodic_loss=zeros(expiry,1); % stores the accumulated loss at each
                               % payment date
out_capital=zeros(expiry,1); % outstanding tranche capital at each
                             % payment date
fee=zeros(expiry,1);
total_fee=0;
indicator=0;
c=0;
% calculate the total loss in the k-th simulation
 for i=1:num
     if def_time(1,i)<expiry % if the simulated default time for the generic
                             % credit is < than the CDO maturity, there is
                             % a loss
         loss(i)=(1-rec)*capital;
         tot_loss=tot_loss+loss(i); % sum the individual losses
    end
end

%% DEFAULT LEG SIMULATION %%

% if the loss is below the lower treshold C there's no default payment
if tot_loss<C
    PV_def=0;
% if the loss is above C and below D there's a default payment
elseif tot_loss>C & tot_loss<D
    for i=1:num
        if def_time(1,i)<expiry
            indicator=indicator+loss(i); % we memorize the cumulative
                                         % losses
            if indicator>C % tranche begins to absorbe losses in excess
                           % of C
                if c==0
                    disc_fact_def=0;
                    r=zc_rate;
                    disc_fact_def=(1+r)^(-def_time(1,i)); % discount factor
                                                          % at default
                    PV_def=PV_def+(indicator-C)*disc_fact_def; % only the
                      % loss exceeding C is absorbed (look at the footnote
                      % 32 in the simulation algorithm p. 51 of my paper)
                     c=1;
                 else
                     disc_fact_def=0;
                     r=zc_rate;
                     disc_fact_def=(1+r)^(-def_time(1,i));
                     PV_def=PV_def+loss(i)*disc_fact_def;
                 end
             end
         end
     end
 % if portfolio losses are exceeding D, the C-D% tranche only absorb losses
 % up to D%.
 elseif tot_loss>D
     for i=1:num
         if def_time(1,i)<expiry
             indicator=indicator+loss(i);
```

```
            if indicator>C & indicator<D % check if losses are in the
                                         % C-D% range
                if c==0
                    disc_fact_def=0;
                    r=zc_rate;
                    disc_fact_def=(1+r)^(-def_time(1,i));
                    PV_def=PV_def+(indicator-C)*disc_fact_def;
                    c=1;
                else
                    disc_fact_def=0;
                    r=zc_rate;
                    disc_fact_def=(1+r)^(-def_time(1,i));
                    PV_def=PV_def+loss(i)*disc_fact_def;
                end
            elseif indicator>D % look at footnote 33 p. 52 of my paper
                if c==1
                    disc_fact_def=0;
                    r=zc_rate;
                    disc_fact_def=(1+r)^(-def_time(1,i));
                    absorbed_loss=D-(indicator-loss(i)); % look at footnote
                                                         % 33 p. 52 of my
                                                         % paper
                    PV_def=PV_def+(absorbed_loss*disc_fact_def);
                    c=2;
                end
            end
        end
    end
end


%% PREMIUM LEG SIMULATION %%

for i=1:expiry
    periodic_loss(i)=0;
    for j=1:num
        if def_time(1,j)<i
            % calculated the accumulated portfolio losses
            periodic_loss(i)= periodic_loss(i)+(1-rec)*capital;
        end
    end
    out_capital(i)=min(max(D-periodic_loss(i),0),D-C); % oustanding capital
                                                       % at each payment
                                                       % date
    fee(i)=((1+zc_rate)^(-i))*out_capital(i);
    PV_premium=PV_premium+fee(i); % DV01

end
```

gaussian_time.m

```
function def_times1=gaussian_time(R,num,n)

% This function generates pseudo default stopping times (generated by a
% multinomial gaussian copula with correlation matrix "R") for the "n"
% obligors included in the basket. "num" stands for the   of simulations.
% We call the output of the function "pseudo" cause we still have to
% divide the result of the function by the specific hazard rate. But given
% the relationship h=S/(1-Rec), the hazard rate will vary at each recovery
% loop, so we will perform the ratio in the CDO_tranche.m function.
```

```
% Now we use the simulation algorithm for generating normal variates from
% gaussian copula suggested by ELM(01), pag. 26

x1=zeros(num,n);
y=zeros(num,n);

A=chol(R);
clear R;
z=randn(num,n);
for i=1:num
y(i,:)=z(i,:)*A;
end
clear A;
clear z;

Gauss_U1=normcdf(y);          % 0-1 variates normally distribuited
clear y;
x1=-log(Gauss_U1);
clear Gauss_U1;

def_times1=x1; % we do not divide by the hazard rate cause it depends on
               % the recovery rate at each loop
```

设一每年5%的平直零曲线和一具有如下特性的5年期债务抵押债券：

票据份额	连接点
股权份额	0%～3%
中间份额	9%～14%
优先份额	14%～100%

以下为图形输出. 图4.15为股权份额，图4.16为中间份额. 图4.17展示了抵押资产为同质资产（数量为$n=100$参考实体，名义金额均为$N_i=100$）的定价的数值计算结果. 其对应单名信用违约掉期曲线恒为每年$s=150$个基点而两两之间的相关性恒为$\gamma_{i,j}=\gamma$.

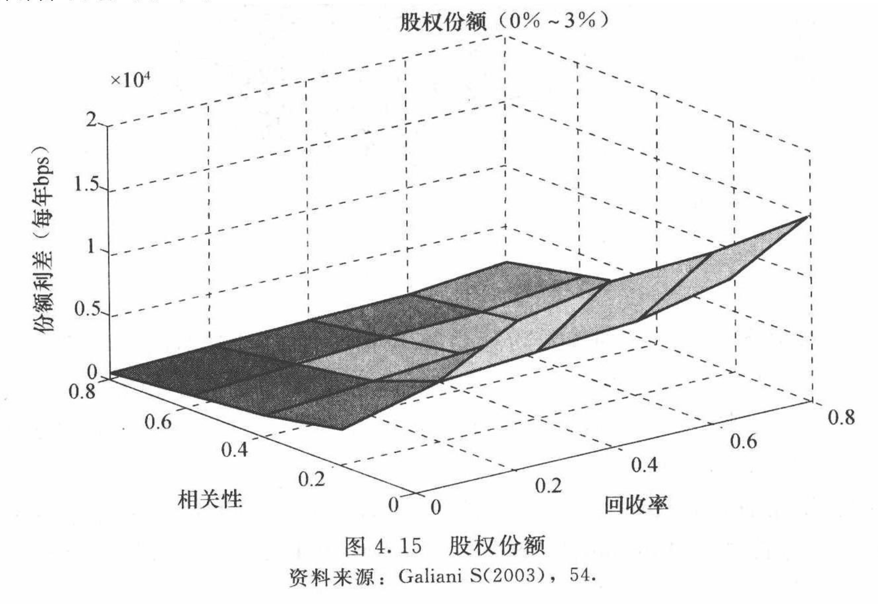


图4.15　股权份额

资料来源：Galiani S(2003)，54.

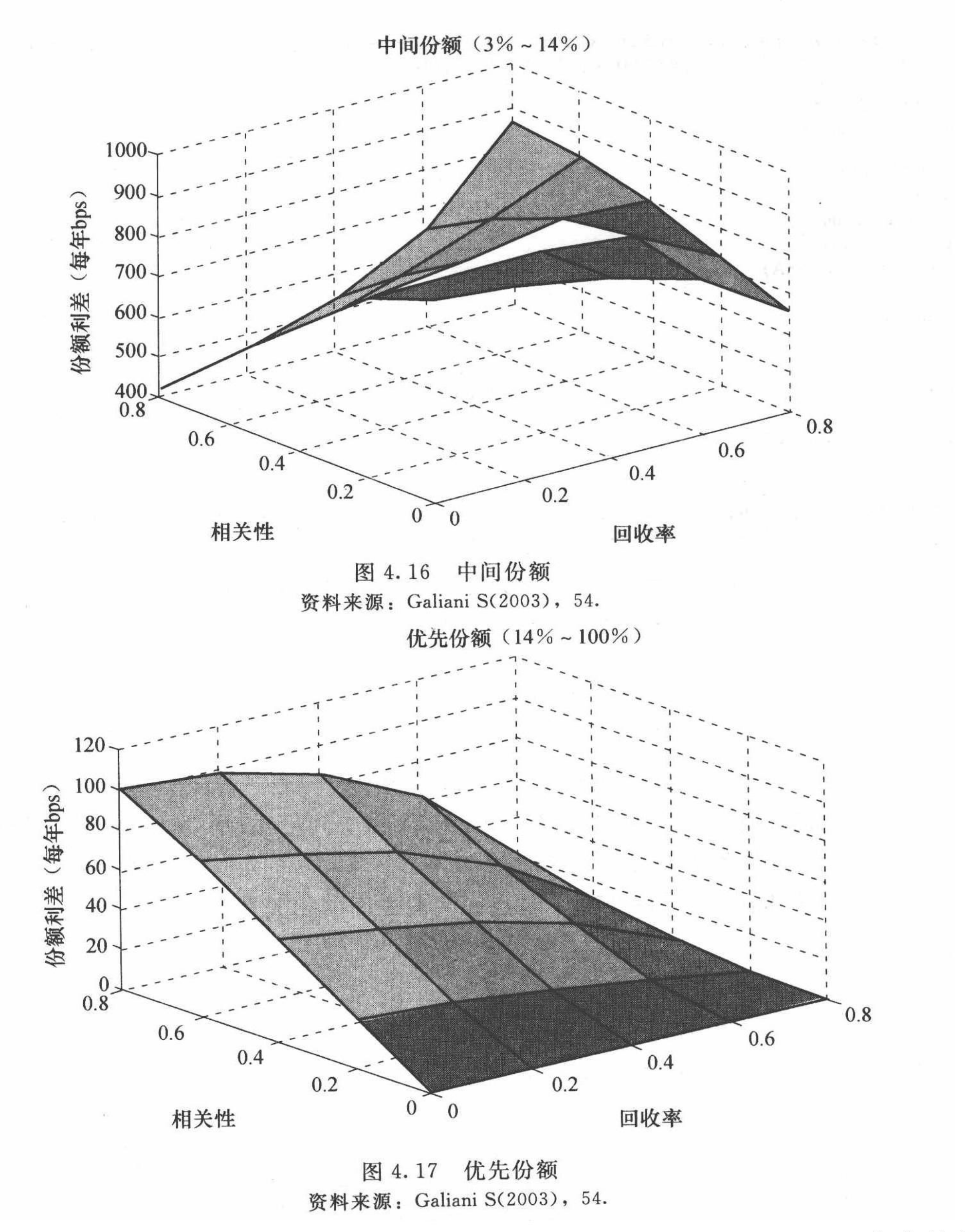


图 4.16 中间份额

资料来源：Galiani S(2003)，54.

图 4.17 优先份额

资料来源：Galiani S(2003)，54.

Gailiani(2002)中指出债务抵押债券份额价格和相关性之间有不同的联系：优先份额表现为价格和相关性正相关，中间份额对于相关性的关系和股权份额对于相关性的关系很相似，尤其是回收率低于 60%的情况. 尽管这样，中间份额和回收率的相关性显得简单明了，相关性越高，其关联形式就和初级份额更相似.

就回收率相关性而言，尽管优先份额的利差单调递减，而对于股权份额来说就恰恰相反. 股权份额的行为首先在 Boscher 和 Ward（2002）中提出，虽然其行为方式乍看起来似

乎违反直觉，但是通过对不同回收率假设下抵押资产组合的亏损分布形状进行分析，可以对此作出合理的解释.为了这个目标，从等式（4.14）出发，可以计算出前两个亏损分布，从而得到：

$$\begin{aligned} E[L(t)] &= E\Big[\sum_{i=1}^{n} L_i Q_i(t)\Big] = \sum_{i=1}^{n} L_i E[1_{\tau_i<t}] \\ &= \sum_{i=1}^{n}(1-R_i)N_i F_i(t) \end{aligned} \tag{4.19}$$

$$V[L(t)] = \sum_{i=1}^{n} V[L_i(t)] + \sum_{i=1}^{n}\sum_{\substack{j=1\\ i\neq j}}^{n} r_{ij}\sqrt{V[L_i(t)]V[L_j(t)]} \tag{4.20}$$

其中，

$$V[L_i(t)] = (1-R_i)^2 N_i^2 F_i(t)(1-F_i(t))$$

对于同质抵押资产（每个参考实体的名义金额同为 n，回收率同为 R），在两两相关性 $r_{ij}=r$ 为恒定值、信贷违约曲线平坦[32]的情况下，对于所有的 i，$j\in\{1, 2, \cdots, n\}$ 都有 $V[L_i(t)]=V[L_j(t)]$；这也就意味着可以将等式（4.20）写成如下形式：

$$V[L_i(t)] = nV[L_i(t)] + n(n-1)rV[L_i(t)]$$

接下来我们定义预期外亏损 UL 为组合亏损方差的平方根：

$$\begin{aligned} UL(t) &= \sqrt{nV[L_i(t)] + n(n-1)rV[L_i(t)]} \\ &= \sqrt{n+n(n-1)r}(1-R)N\sqrt{F(t)(1-F(t))} \end{aligned} \tag{4.21}$$

很明显，亏损分布的偏离度（也就是 UL）对于相关性 r 是一个增函数.考虑到回收率的因素，令 $Q=\sqrt{n+n\ (n-1)\ r}\,N=Q$，并将式（4.12）关于回收率 R 进行微分，从而得到

$$\begin{aligned} \frac{\partial UL(t)}{\partial R} &= Q\Big[\frac{\partial(1-R)}{\partial R}\sqrt{F(t)(1-F(t))} + (1-R)\frac{\partial\sqrt{F(t)(1-F(t))}}{\partial R}\Big] \\ &= Q\Big[-\sqrt{F(t)(1-F(t))} + \frac{st(1-R)^{-1}(1-F(t))(1-2F(t))}{2\sqrt{F(t)(1-F(t))}}\Big]. \end{aligned}$$

令 $W=st/(1-R)$，通过简单的代数运算，有：

$$\frac{\partial UL(t)}{\partial R} = Q\sqrt{1-F(t)}\Big[-\sqrt{F(t)} + W\frac{1-2F(t)}{2\sqrt{F(t)}}\Big]$$

对于 $R<1$ 和 $s>0$ 的情况，方括号内的值为负，从而意味着抵押资产组合的预期外亏损（标准差）是一个关于回收率单调递减的函数.

图 4.18 画出了债务抵押债券抵押资产组合（$n=100$）对于不同回收率的亏损分布函数.

和预料的一样，回收率相当于亏损分布函数的一个比例定位参数，将亏损分布函数从中心向中心偏左的方向平移[33]，并且减小了亏损分布函数的标准差（在回收率增大的情况下).这显然影响了债务抵押债券的定价，从而使得初级份额的利差增大而优先份额的利差减小.

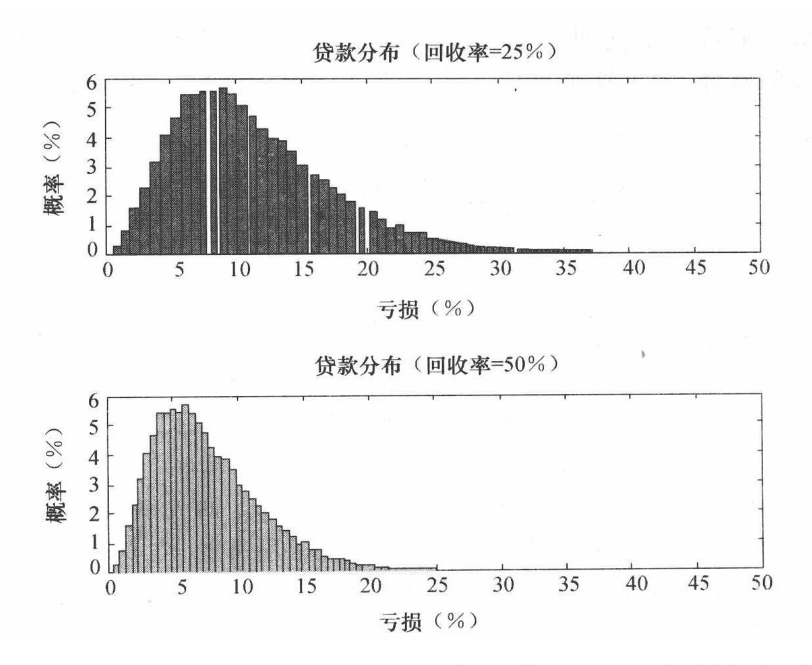


图 4.18 通过高斯 copula，用抵押资产组合模拟亏损（$r=0.1$，恒量 $h=k=3\%$）

资料来源：Galiani S(2003)，57.

4.10 C++中的债务抵押债券定价

CDO.h

```
#ifndef _CDO__
#define _CDO__


#include "datecl.h"
#include <vector>
#include <map>
#include <iostream>
#include "MatrixUtil.h"

#define N 1
#define NUM_TRANCHES 3
#define NUM_REF 100
#define numSim 10000
#define MATURITY 5
#define ZC 0.05
#define ENTITY_NOTIONAL 10000
#define SPREAD 150

class Tranche
{
 public:
        Tranche(double C, double D, string desc) :
          lowerAttachment(C*ENTITY_NOTIONAL*NUM_REF),
        upperAttachment(D*ENTITY_NOTIONAL*NUM_REF), desc_(desc) {}
        vector<double> cash_Flow(int expiry, vector<double> defTime,
```

```
    double rec, double capital,double C, double D, int numReference)
{
        vector<double> results;
        double PV_def=0;
        double PV_premium=0;
        // loss given default for each credit
        double* loss = new double[numReference];
        double tot_loss = 0;        // cumulative portfolio loss
        // stores the accumulated loss at each payment date
        double* periodic_loss = new double[numReference];
        // outstanding tranche capital at each payment date
        double* out_capital = new double[numReference];
        double* fee = new double[numReference];
        double total_fee = 0;
        double indicator = 0;
        double disc_fact_def = 0;
        double absorbed_loss = 0;
        double r = 0.0;
        int c = 0;
        int j = 0;
        int num = N;

        // calculate the total loss in the k-th simulation
        for (int i = 0; i < numReference; i++)
        {
        // if the simulated default time for the generic credit
        // is < than the CDO maturity, there is a loss
           if (defTime[i] < expiry)
           {
                loss[i] = (1- rec)*capital;
                // sum the individual losses
                tot_loss = tot_loss + loss[i];
           }
        }
        // DEFAULT LEG SIMULATION %%
        // if the loss is below the lower treshold C there's no
        // default payment
        if (tot_loss < C)
                PV_def=0;   // if the loss is above C and below D
                            // there's a default payment
         else if ((tot_loss >= C) && (tot_loss < D))
         {
            for (i= 0; i < numReference; i++)
            {
                 if (defTime[i] < expiry)
                         {
                            // we memorize the cumulative losses
                    indicator=indicator+loss[i];
                            // tranche begins to absorbe losses in
                            // excess of C
                    if (indicator > C)
                    {
                        if (c == 0)
                        {
                            disc_fact_def=0;
                            r = ZC;
                            // discount factor at default
                            disc_fact_def= exp(-defTime[i]*r);
                               // only the loss exceeding C is
```

```
                    // absorbed (look at the footnote
                    // 32 in the simulation algorithm
                    // p.51 of my paper)
                    PV_def=PV_def+(indicator-C)*
                      disc_fact_def;
                    c = 1;
                 }
                 else
                 {
                    disc_fact_def=0;
                    r= ZC;
                    disc_fact_def= exp(-defTime[i]*r);
                    PV_def=PV_def+loss[i]*disc_fact_def;
                 }
              }
           }
      }
   }
   // if portfolio losses are exceeding D, the C-D% tranche
   // only absorb losses up to D
   else if (tot_loss > D)
   {
      for (i = 0; i < numReference; i++)
      {
           if (defTime[i] < expiry)
           {
             indicator= indicator + loss[i];
                 // check if losses are in the C-D% range
             if ((indicator > C) && (indicator < D))
                 {
                if (c == 0)
                   {
                     r= ZC;
      disc_fact_def = exp(-defTime[i]*r);
      PV_def=PV_def+(indicator-C)*disc_fact_def;
      c=1;
                 }
            else

                 {
                      r= ZC;
                      disc_fact_def= exp(-defTime[i]*r);
                 PV_def=PV_def+loss[i]*disc_fact_def;
              }
           }
                 // look at footnote 33 p. 52 of my paper
            else if (indicator > D)
            {
                 if (c == 1)
              {
                      r= ZC;
                      disc_fact_def= exp(-defTime[i]*r);
                          // look at footnote 33 p. 52
                          // of my paper
                      absorbed_loss = D -
                        (indicator-loss[i]);
                      PV_def=PV_def+(absorbed_loss*
```

```
                                          disc_fact_def);
                                        c=2;
                                   }
                              }
                         }
                    }
               }
               results.push_back(PV_def);


          // PREMIUM LEG SIMULATION

          for (i= 0; i < expiry; i++)
          {
                    periodic_loss[i]=0;
                    for (j= 0; j < numReference; j++)
                    {
                    if (defTime[j] < i)
                              // calculated the accumulated portfolio losses
                              periodic_loss[i] = periodic_loss[i]+
                                (1-rec)*capital;
                    }
                    // oustanding capital at each payment date
                    out_capital[i] = min(max(D-periodic_loss[i],0),D-C);
                    fee[i]= exp(-ZC*i)*out_capital[i];
                    PV_premium=PV_premium+fee[i]; // DV01
          }

          results.push_back(PV_premium);
          results.push_back(indicator/(ENTITY_NOTIONAL*NUM_REF));

          return results;
          }
          void setExpectedLoss(double loss) { expectedLoss_ = loss; }
          double getExpectedLoss() const { return expectedLoss_; }
          double getUpperAttachment() { return upperAttachment; }
          double getLowerAttachment() { return lowerAttachment; }
          void setSpread(double spread) { spread_ = spread; }
          double getSpread() const { return spread_; }
          void setDesc(string desc) { desc_ = desc; }
          string getDesc() const { return desc_; }
private:
          double expectedLoss_;
          double upperAttachment;
          double lowerAttachment;
          double recovery;
          double spread_;
          string desc_;
          double totalLoss;      // cumulative portfolio loss
          double out_capital;    // outstanding tranche capital at each
                                 // payment date
          double periodic_loss;  // stores the accumulated loss at each
                                 // payment date
};

class CDO
{
 public:
```

```
CDO() : T(MATURITY), numReference_(NUM_REF) { }
double priceTranche(int numReference)
{
        MatrixUtil mu;
        double C = 0.0;
        double D = 0.0;
        double spread = (double) SPREAD/10000;
        Array2D<double> R1(numReference,numReference);
        int size = R1.dim1();
        std::cout << "pool size = " << size << endl;
        vector<double> defaultTime;
        vector<double> normaldev;
        vector<double>::iterator iter;
        vector<double> val;
        map<double,int> expMap;

        double y = 0;
        double tau = 0.0;
        double fees = 0.0;
        double loss = 0.0;
        double PV_default = 0.0;
        double* dev = new double[numReference];
        int rec_cycle = 0;

        Array1D<double> hazard(N);
        Array1D<double> R(N);
        Array1D<double> expectLoss(N);
        Array1D<double> recovery(N);
        Array2D<double> corr(numReference,numReference);
        Array2D<double> S_fees(N,NUM_TRANCHES);
        Array2D<double> S_default(N,NUM_TRANCHES);
        Array2D<double> M_fees(N,NUM_TRANCHES);
        Array2D<double> M_default(N,NUM_TRANCHES);
        Array2D<double> totLoss(N,NUM_TRANCHES);
        Array2D<double> expLoss(N,NUM_TRANCHES);
        //std::cout.precision(5);

        for (int j = 0; j < N; j++)
        {
        // start the loop for each of the three tranches
                for (int u = 0; u < NUM_TRANCHES; u++)
                {
                    S_fees[j][u] = 0;
                    S_default[j][u] = 0;
                    M_fees[j][u] = 0;
                    M_default[j][u] = 0;
                    totLoss[j][u] = 0;
                }
                hazard[j] = 0.03;   // hazard rate
                recovery[j] = 0.40; // recovery rate
        }

        for (int R_cycle = 0; R_cycle < N; R_cycle++)
        {
                for (int xx = 0; xx < numReference; xx++)
                {
                    for (int yy = 0; yy < numReference; yy++)
                {
```

```
                    if (xx == yy)
                        corr[xx][yy] = 1.0;
                    else
                    {
                        corr[xx][yy] = 0.2*(R_cycle+1);
                          // populate the correlation
                          // matrix
                        corr[yy][xx] = 0.2*(R_cycle+1);
                     }
                     R1[xx][yy] = corr[xx][yy];
                }

        }

        for (int i = 0; i < numSim; i++)
        {
            defaultTime.clear();
            defaultTime.empty();
            defaultTime = mu.genCholesky4(R1);

            for (int j = 0; j < N; j++)
            {
                int l = 0;
                for (iter = defaultTime.begin();
                  iter != defaultTime.end(); iter++)
                {

                      defaultTime[l] = (double)
                        *iter/hazard[j];
                      l++;
                 }
                // tau = *iter/hazard[expMap[*iter]];
                // std::cout << "tau = " << tau << endl;
                // generate the vector of default time by
                // dividing by the corresponding
                // hazard rate

                sort(defaultTime.begin(),
                  defaultTime.end());
                // start the loop for each of the three
                // tranches
                for (int u= 0; u < NUM_TRANCHES; u++)
                {
                // calculate the premium and default leg
                C = tranche[u].getLowerAttachment();
                D = tranche[u].getUpperAttachment();
                val= tranche[u].cash_Flow(MATURITY,
                  defaultTime,recovery[j],ENTITY_NOTIONAL,
                  C,D,numReference);
                iter = val.begin();
                PV_default = *iter;
                iter++;
                fees = *iter;
                iter++;
                loss = *iter;
                totLoss[j][u] = totLoss[j][u] + loss;
                S_fees[j][u] = S_fees[j][u]+fees;
                S_default[j][u] = S_default[j][u]+
                  PV_default;
```

```
                    }
                }
                defaultTime.empty();
                defaultTime.clear();
            }

            for (int rec_cycle = 0; rec_cycle < N; rec_cycle++)
            {
                for (int u = 0; u < NUM_TRANCHES; u++)
                {
                    // average DV01
                    M_fees[rec_cycle][u]=
                      S_fees[rec_cycle][u]/numSim;
                    // average default leg
                    M_default[rec_cycle][u]=S_default[rec_cycle]
                      [u]/numSim;
                    expLoss[rec_cycle][u] = totLoss[rec_cycle]
                      [u]/numSim;
                }
            }

            for (rec_cycle = 0; rec_cycle < N; rec_cycle++)
            {
        for (int u = 0; u < NUM_TRANCHES; u++)
                {
            spread= ((M_default[rec_cycle][u])/(M_fees
             [rec_cycle][u]))*10000;
                // B\E spread for the 0-3% tranche
                loss = expLoss[rec_cycle][u];
                tranche[u].setSpread(spread);
                tranche[u].setExpectedLoss(loss);
                    std::cout << tranche[u].getDesc() << " " <<
             tranche[u].getSpread() << " " <<
                    tranche[u].getExpectedLoss() << endl;

                }
            }
        }
        delete [] dev;

        return 0.0;

        }
        void addTranche(Tranche t) { tranche.push_back(t); }
        int getNumTranches() { return numTranches_; }
        int getNumReferences() { return numReference_; }
        virtual ~CDO() { }
  private:
        //Date maturity;
        vector<Tranche> tranche;
        int T;
        int numReference_;
        int numTranches_;
};

#endif
```

主函数是：

Main.cpp

```
#include <strstream>
#include <iostream>
#include <fstream>
#include "Basket.h"
#define SIZE_X 1000
using namespace std;

void main()
{
        CDO cdo;

        std::cout << "Pricing CDO..." << endl;
        Tranche tranche1(0,0.03,"equity");
        Tranche tranche2(0.03,0.14,"mezzanine");
        Tranche tranche3(0.14,1.00,"senior");


        cdo.addTranche(tranche1);
        cdo.addTranche(tranche2);
        cdo.addTranche(tranche3);
        cdo.priceTranche(100);
}
```

范例 1

假设我们现在要为一个债务抵押债券的优先份额、中间份额和股权份额定价，而这些份额的上下限如下表所示：

	下　限	上　限
优先份额	14%	100%
中间份额	3%	14%
股权份额	0%	3%

为了简化，我们假设有组合中的资产同质，有 100 位借款人并且违约率均为 3%. 运行 10 000次蒙特卡罗模拟仿真得到下表：

	价格 (bps)	标准差
优先份额	52.76	0.064 2
中间份额	1 372.9	0.111 5
股权份额	4 672.1	0.113 7

4.11 债务抵押债券的抵押债券 (CDO^2) 定价

Li 和 Liang(2005) 中给出了有 m 个子债务抵押债券或者标的债务抵押债券、总共信贷数目为 N 的 CDO^2 的定价公式. 对于每个子债务抵押债券，我们可知在时刻 t 亏损累积为：

$$L_B^N(t) = \sum_{k=1}^{n} N_i^k(1-R_i)1_{\{\tau_i<t\}} \tag{4.22}$$

其中 N_i^k 为位于第 k 个子债务抵押债券中的第 i 笔信贷的名义金额，τ_i 为第 i 笔信贷的违约时

刻. 该份额亏损 $\overline{L}_B^k$，即第 k 份子债务抵押债券中上下限分别为 A_L^B 和 A_U^B 的份额的亏损为

$$\overline{L}_B^k = \max(L_B^k(t) - A_L^B, 0) - \max(L_B^k(t) - A_U^B, 0) \tag{4.23}$$

CDO² 或者母债务抵押债券中上下限分别为 A_L^M 和 A_U^M 的份额的亏损为：

$$\overline{L} = \max\Big(\sum_{k=1}^{n} L_B^{\bar{k}}(t) - A_L^M, 0\Big) - \max\Big(\sum_{k=1}^{n} L_B^{\bar{k}}(t) - A_U^M, 0\Big) \tag{4.24}$$

如同先前的收益所示，CDO² 定价中所依赖的收益函数就是所有标的子债务抵押债券组合的联合亏损分布.

4.12 CDO 和 CDO² 的快速亏损计算

令 $q_i(t \mid Y_M) = Pr[\tau_i \leqslant t \mid Y_M]$ 表示时刻 t 以前第 i 笔信贷的条件边际违约率，其中 Y_M 为条件状态变量. 然后，将总的亏损分布视为所有相互独立的变量的和，有几种方法可以用来计算这个总亏损分布，如傅里叶变换、递归法或者条件正态近似法.[34] 总亏损 $L \mid Y_M$ 的条件均值和方差分别是：

$$M_v = \sum_{i=1}^{n} N_i(1 - R_i) \cdot q_i(t \mid Y_M)$$

$$\sigma_v^2 = \sum_{i=1}^{n} N_i^2(1 - R_i)^2 \cdot q_i(t \mid Y_M)(1 - q_i(t \mid Y_M))$$

Li 和 Liang(2005) 中使用了条件正态近似法. 假设这个正态分布与先前计算结果有着相同的均值和方差，这种方法用正态分布来近似总的条件亏损分布.[35] 使用正态分布的原因是中心极限定理阐释了随着独立分布数的增加，它们的和的分布（并非相同的分布）越来越趋向于正态分布.

Li 和 Liang(2005) 中使用这种条件正态近似的方法，给出了某个份额的条件期望亏损分布的易于计算的闭合型公式：

$$\begin{aligned} E[L^T(t) \mid v] = {} & (M_v - A_L^T)\Phi\Big(\frac{M_v - A_L^T}{\sigma_v}\Big) + \sigma_v \cdot \phi\Big(\frac{M_v - A_L^T}{\sigma_v}\Big) \\ & - (M_v - A_U^T)\Phi\Big(\frac{M_v - A_U^T}{\sigma_v}\Big) + \sigma_v \cdot \phi\Big(\frac{M_v - A_U^T}{\sigma_v}\Big) \end{aligned}$$

其中 ϕ 是一维正态分布概率密度函数. 在计算出了条件期望亏损之后，可以简单地通过对于共同因子 Y_M 进行积分获得无条件期望亏损：

$$E[L^T(t)] = \int_{-\infty}^{\infty} E[L^T(t) \mid y] \cdot \phi(y)\,\mathrm{d}y$$

对于每一个股权份额，可以用以下方法来保存整个组合的预期亏损，从而得到一个更准确的近似. 在股权份额收益函数中，我们允许使用条件正态近似方法时出现亏损为负的情况. 因此上限为 A_U^T 的股权份额的亏损可以表达如下：

$$L_{\text{equity}}^T(t) = L(t) - \max(L(t) - A_U^T, 0)$$

股权份额的条件期望亏损为：

$$E[L_{\text{equity}}^T(t) \mid v] = M_v - (M_v - A_U^T)\Phi\Big(\frac{M_v - A_U^T}{\sigma_v}\Big) - \sigma_v \cdot \phi\Big(\frac{M_v - A_U^T}{\sigma_v}\Big)$$

这一方法被证明在指数股权份额大于3%的情况下效果良好.[36]此外，我们还可以用反高斯分布来近似股权份额的情况，因为反高斯分布只能取正值.

类似地，合成 CDO^2 的价格依赖于所有标的子债务抵押债券资产组合的联合亏损分布.基于正态变量 Y_M，我们可以计算出条件亏损变量 $L_K/Y_M(k=1,\cdots,n)$ 的均值和协方差：

$$\mu_k^v = \sum_{i=1}^{n} N_i^k(1-R_i).\, q_i(t\,|\,Y_M)$$

$$\sum_{k,k'}^{v} = \sum_{i=1}^{n} N_i^k N_i^{k'}(1-R_i)^2.\, q_i(t\,|\,Y_M)(1-q_i(t\,|\,Y_M))$$

我们可以从中发现两个子资产组合的亏损分布的条件协方差取决于这两个资产组合的重合度.[37]即使每笔信贷基于一个共同条件因子的违约率彼此之间是相互独立的，两个资产组合中资产的重合会造成两个组合亏损分布的高度相关性.条件正态近似法使用多正态分布来近似条件亏损分布.多正态分布均值和协方差矩阵与前面计算的相同.由于 CDO^2 的亏损是标的债务抵押债券亏损变量的一个函数，并且在条件正态分布的前提下，可以用多维积分来计算其条件期望亏损：

$$E[\overline{L}_t\,|\,v] = \int\cdots\int \overline{L}(L_b^1,\cdots,L_b^n)\,\mathrm{d}L_b^1\,\mathrm{d}L_b^2\cdots\mathrm{d}L_b^n$$

该积分的维数和标的债务抵押债券的数量相等.使用蒙特卡罗模拟来计算这个多维积分.在得到了条件期望亏损之后，非条件亏损期望值可以通过对共同因子 Y_M 进行积分得到：

$$E[\overline{L}_t] = \int_{-\infty}^{\infty} E[\overline{L}_t\,|\,y]\cdot n(y)\,\mathrm{d}y$$

其中，$n(y)=\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-y^2}$ 为标准正态密度函数.

对于债务抵押债券和 CDO^2 的定价，需要知道时间轴上不同时间的组合亏损分布.从上限为 A 的股权份额出发，该份额到时刻 t 的累积亏损 $L_T(t)$ 为：

$$L_T(t) = \max(L(t),0) - \max(L(t)-A,0) = L(t) - \max(L(t)-A,0)$$

Matlab 中计算债务抵押债券份额亏损的快速算法

在使用数值快速积分的高斯因子模型中，Pavel Okenuv(2005) 介绍了一种计算贷款组合债务抵押债券的期望损失的 Matlab 算法实现.

gs_loss.m

```
% This implements one factor Gaussian model.
% [loss]=gsloss(L,w,p,a,d,N)
% L = exposures,as fraction of total
% portfolio, taking into account the recovery rate
% Example: loan 1 is 0.01 fraction of the total portfolio, recovery
% rate is 40% then L(1)=0.01*(1-0.4)
% w = loading factors
% p = default probabilities
% a = attachement point
% d = detachment point
% N = number of names in the portfolio
% loss = expected tranche loss as percentage of the portfolio nominal
% expressed in basis points

function [loss]= gsloss(L,w,p,a,d,N)
```

```
Pi=3.14159265;
% This integration routine serves as an example only. We highly
% recommend that the user should replace it with his/her favorite high
% order routine.
% Set integration parameters
IP=200; % number of points used by integration routine
BD=10;   % integration interval [-BD,BD]
z=linspace(-BD,BD,IP+1); % creates equally spaced grid for integration
% Loss computation
loss=0;

for i=1:IP

  m=0; % m conditional mean
v=0; % v conditional variance
for k=1:N
   pm=normcdf((norminv(p(k))-w(k)*z(i))/sqrt(1-w(k)*w(k)));
m=m+L(k)*pm;
v=v+L(k)*L(k)*pm*(1-pm);
  end

  s=sqrt(v); % s conditional standard deviation
  temp=m*(normcdf((d-m)/s)-normcdf((a-m)/s));
  temp=temp+s*1/sqrt(2*Pi)*(exp(-(a-m)^2/(2*s^2))-exp(-(d-m)^2/(2*s^2)));
  temp=temp+(d-a)*normcdf((1-m)/s)-d*normcdf((d-m)/s)+a*normcdf((a-m)/s);
  loss=loss+temp*1/sqrt(2*Pi)*exp(-z(i)^2/2)*2*BD/IP;
end

loss=loss*10000; % express as basis points
```

尾注

1. Picone D, pg. 2.
2. Id. pg. 2.
3. Gibson M (2004), pg. 1.
4. Id. pg. 1.
5. Id.
6. Hull J and White A (2004), pg. 17.
7. Id. pg. 17.
8. Gibson M, pg. 1.
9. 根据 Goodman(2002) 中的介绍，第一份合成债务抵押债券于 1997 年由瑞士银行的“Glacier Finance Ltd.”和摩根大通的“BISTRO”发行.
10. Gibson M (2004), pg. 2.
11. Id. pg. 2.
12. Id. pg. 3.
13. Id. pg. 2.
14. Id. pg. 2.
15. Id. pg. 3.
16. Gibson M (2004), pg. 3.
17. Id.
18. 37%对应了穆迪所计算的 6 年累积违约率.
19. 如 Merton (1974) 中所述，当一家公司的资产值跌破一定阈值的时候就会发生违约情况.

因此，公司价值就是提取信用事件的过程. 和其他信贷风险模型不同，通过引入每个信用评级向另一个信用评级转移的阈值，CreditMetrics 也能够处理借贷人的信用资质转移问题. 该模型的一个关键特征就是其完整的亏损分布无法通过数学分析的方法来求解而只能通过蒙特卡罗方法来模拟. 为了能够运行模拟，需要首先确定以下参数：

- 对于每个借贷人，位于时间轴末端可能出现的价值必须确定. 在没有违约发生的情况下，该价值等于贴现现金流之和. 在违约的情况下，它就等于该违约的回收率.
- 基于资产回报率为标准正态分布的假设，我们可以将根据历史数据预测的违约率转化为每个借贷人的违约阈值.
- 借贷人之间的违约相关性.

20. 参见 Li(2000)；Boscher and Ward (2002)；Andersen，Sidenius，and Basu (2003)；以及 Hull and White (2003) .
21. 在一台 AMD Athlon 2. 2 GHz，1GB 内存的 PC 上，Gibson (2004) 中利用 Scilab 软件包，花费 10 秒计算了 $N=100$ 的资产组合的无条件亏损分布 (http://www. scilab. org) .
22. Gibson M(2004)，pg. 18.
23. Gibson M(2004)，pg. 18.
24. Id. pg. 18.
25. Lehnert N，Altrock R，Rachev S，Trück S，and Wilch A (2005)，pg. 10.
26. Id. pg. 10.
27. Id. pg. 10.
28. Id. pg. 10.
29. McGinty and Ahuluwalia(2004) .
30. 假设 30/360 的记日方法，$\alpha=1$ 为每笔支付的年支付频率，$\alpha=1/2$ 为半年支付频率，$\alpha=1/4$为每季支付频率.
31. Matlab 代码由 Stefano Galiani 编写.
32. 平坦利差曲线的假设暗含了利率、合约利差、风险率和回收率之间的关系，也就是：

$$h=\frac{s}{1-R}$$

完整的证明参考 Meneguzzo 和 Vecchiato (2002)，44～54 页. 此外，基于此性质，可以将违约时间分布函数表示为：

$$F(t)=1-\exp(-ht)=1-\exp\left(-\frac{s}{1-R}t\right)$$

33. 基于该假设，这个断言可以通过对回收率 R 微分来简单地证明，从而得到：

$$\frac{\partial E[L(t)]}{\partial R}=-nN\left[1-\left(1+\frac{st}{1-R}\right)\exp\left(-\frac{st}{1-R}\right)\right]$$

对于 $s>0$ 和 $R<1$，其值为非正.

34. Li and Liang(2005)，pg. 13.
35. Li and Liang(2005)，pg. 13.
36. Id. pg. 14.
37. Id. pg. 14.

第5章 信用衍生品

本章讨论信用衍生品——这些衍生品使得对信用风险进行有效的转移和再分配成为可能.例如信用违约掉期这样的信用衍生品，能够使金融机构的贷款资产组合和资产负债表免受标的资产的信用风险以及其他信用相关事件的影响，如信用息差的扩大、信用评级的下降、借贷人的违约.举个例子来说，信用衍生品可以使银行转移其信用风险，并将一大部分管理资本从资产表中取出，但仍然能从事信用相关的经营活动.

过去5年来，信用衍生品市场经历了极大的成长，从1995年的几乎没有到如今的总市值达到1万亿美元.大部分增长归功于信用衍生品比现金具有越来越多的优势.例如，一个标准的信用违约掉期可以通过一种现金债券和回购市场来进行复制.去除套利机会后，与标准的完全融资的现金债券相比，使用信用违约掉期这样一个未完全融资的产品来复制信用风险，显得更有效率、花费更低.在引入未融资产品后，信用衍生品头一次将融资问题从信用风险中分离出去.[1] 这使得信用衍生品更倾向于被那些有着高融资成本但能够较经济地杠杆化信用风险的人们所使用.[2]

一些形式更为奇异的信用衍生品能够“将个别资产或个别资产组所承担的信用风险分割开，并重新分布成更为密集或更为分散的能够满足不同风险偏好的投资者的产品”.[3] 比如说，寻求收益的投资者可以购买债务抵押债券（CDO）和份额化的资产组合违约掉期的先损份额产品来杠杆化信用风险和信用收益.而对风险更为厌恶的投资者可以买入风险更低、收益更低的后损份额产品.

在本章中，我们详细研究了多种信用衍生品以及信用衍生品的定价模型.5.1节讨论了信用违约掉期的工作原理和结构.5.2节回顾了信用违约掉期的记日规则.5.3节讨论了信用违约掉期的通用定价原则.5.4节回顾了风险率函数，这个函数被用来计算违约概率.5.5节讨论了建立违约时间模型和违约数模型的泊松过程和Cox过程.5.6节讨论了使用确定性强度模型来为信用违约掉期进行定价.5.7节将风险率校准和Bloomberg真实市场数据进行对照.5.8节讨论了信用曲线的建立和校准.5.9节介绍多标的物信用衍生品，这类衍生品有一个一揽子信用违约掉期和指数.5.10节提供了一种基于5.6节中介绍的算法来实现的信用违约掉期定价的Matlab程序.5.11节介绍在C++上实现的一揽子信用违约掉期和信用指数定价.5.12节讨论了信用联系票据.

5.1 信用违约掉期

信用违约掉期是供信用风险的套期保值者和投资者使用的一种标准的未融资的信用衍生品.信用违约掉期是在保护的买方和保护的卖方之间达成的一个掉期协议.保护的买方（做空信用）向保护的卖方（做多信用）购买，以保护资产（例如由某个参考实体（reference entity）发行的公司债券）违约的风险.保护的买方通常每季支付一次费用，该费用由特定信用违约掉期的利差所决定.

信用违约掉期通常都在3月6月9月12月的20日到期，并采用顺延一天的记日更正方法.大部分流动性强的信用违约掉期都有5年的到期日，名义金额为1 000万美元.

随着某个指定的信用事件发生，比如违约、信用评级下降、破产、无法支付本金或利息、拒付债务和债务在重组，保护的买方都能收到一笔费用，这笔费用用来补偿持有多头该标的信用物带来的损失.比如，假设2004年2月5日，美洲银行（BoA）向福特汽车公司提供了一笔三年期总额1 000万美元的贷款，到期日为2007年2月7日.美洲银行对于福特的信用质量下降从而产生违约有所顾虑，为了进行自我保护，美洲银行可以选择向互换的卖方购买一份三年期（2007年3月20日到期）针对福特（参考债务人）的保护，并将福特公司发行的债券——比如6.75%的2007年优先未保护债券——作为参考资产.[4]

如果在掉期合约的订立日福特公司的利差为156点，那么每个季度美洲银行都要向保护的卖方支付 $10\ 000\ 000 \times 0.25 \times 0.015\ 7 = 39\ 250$ 美元.而实际上保护的买方支付的费用是 $(A/360) \times 10\ 000\ 000 \times 0.015\ 7 = (A/360) \times 157\ 000$ 美元，其中 A 是实际支付日之间的日期数.此外，如果在两个费用支付日之间有信用事件发生，那么保护的买方支付的费用是从前一个支付日起算的违约掉期收益的累计利息部分.

违约费用支付的处理细则是既可以通过现金结算也可以通过实物结算.在现金结算的情况下，卖方支付的费用要么是先前确定的，要么是由信用事件发生之后参考资产的票面价值和回收价值之间的价差确定.在实物结算的情况下，保护的买方向卖方递交违约了的参考证券，而卖方向买方支付证券的票面价值和违约后资产的市场价之间的价差.为了使回收价值能够稳定下来，违约资产的价格通常是根据信用事件发生后14～30天内进行的经销商询价结果来确定.在无法对其进行定价的时候，会在合约中注明允许使用另一个有着相同信用等级并且到期日相近的资产的价格来代替.

如果在信用违约掉期到期日之前，福特公司发生了对其发行的公司债券的违约行为，我们就认为发生了一项信用事件.要达到一个实际的信用事件的标准，违约数额必须超出一定的物质性界线，并且该数额要在宽限期之后仍未得到支付.[5]

假定该债券的回收率为40%.在现金结算的情况下，保护卖方要向美洲银行支付600万美元，而美洲银行要向保护卖方支付直到违约发生日为止的累计违约金额.在实物结算的情况下，美洲银行向保护卖方递交10 000份福特公司2007年6.75%的（违约）债券，名义总金额为1 000万美元（信用违约掉期的名义金额），而作为交换，从保护卖方得到1 000万美元的回报.[6] 由于违约证券的流动性非常低，信用违约掉期合约中往往包含一个以上的债券供交割.因此，美洲银行可以选择具有最低市场价值的债券来进行交割，也就是说，信用违约掉期为其提供了一个“最廉价交割”的期权.

假设参考资产的回收率是40%，图5.1画出了存在或不存在违约时的现金流.

信用违约掉期是一个票面价值产品，因此，它无法对不以票面价值交易的资产的损失进行完全的套期保值.信用违约掉期在资产以低于票面价值进行交易时会造成过度对冲，而在资产溢价交易时造成对冲不足.如果资产没有信用事件发生但是价格下降很多，投资者可以

以较低面值买入保护，也可以使用摊销违约掉期. 随着到期日的临近，对冲的数量分摊到债券的面值上[7].

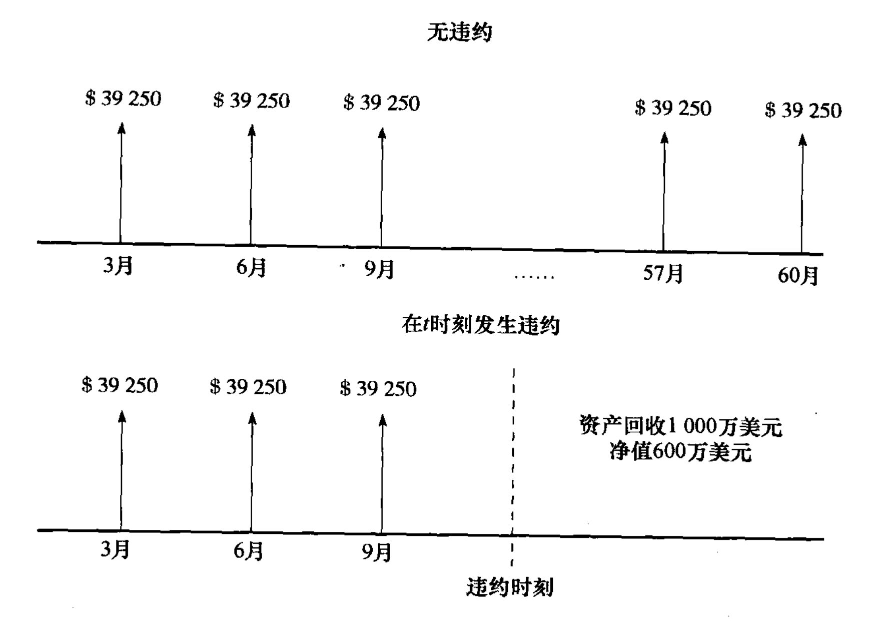


图 5.1　信用违约掉期现金流

根据紧盯市场（mark-to-market）原则，信用违约掉期的价值会随着债券发行者信用质量的改变而改变，这反映在发行者违约掉期息差的改变上，因为紧盯市场的违约掉期必须反映进入一个反向交易所需的花费. 对于保护的买方来说，违约掉期的头寸反映了进入一个与多头头寸到期日相同的空头头寸所需的花费.

令 $S(T)$ 表示现有违约掉期到到期日的息差，$S(0)$ 表示违约掉期在交易发生时的息差，$PV01$ 表示 0 回收率的、一个基点年金的现值，这个年金在信用事件发生后被终止的违约掉期有着相同到期日. 那么紧盯市场的价值为：

$$MTM = (S(T) - S(0)) \times PV01$$

随着时间的推移，信用违约掉期的盯市价由于其到期日的缩短就会下降. 如果发行人的信用质量变差，保护的多头（看空信用）的盯市价就会上升，而如果发行人的信用质量提高，那么保护多头的盯市价会下降.

5.2 信用违约掉期的记日规则

信用违约掉期的执行日称为交易日 t. 第二天，也叫做 $t+1$ 日，称为有效日，$t+3$ 天称为交割日. 通常来说，信用违约掉期交易手册使用一种改良的次日记日法. 使用这种记日法，如果交割日或者支付日是在周末或者节假日，就用下一个工作日来计算. 保护买方的溢价费用以实际日/360 来计算，而保护卖方的溢价费用以实际日/365 来计算.

5.3 信用违约掉期的一般性估值

为了给一个信用违约掉期估值，首先要设定一个利差，从而使得保护买方和卖方之间的现金流净现值为 0. 可以用 Duffie(1998)、Lando(1998)、Duffie 和 Singleton(1999) 中介绍的著名的简化形式框架来对违约掉期进行估值.[8] 与 Merton(1973) 提出的结构模型相反，在一个简化模型下，假设违约时刻以一个外源强度服从一个跳跃过程. 只要这个强度是一个仿射扩散变量的线性函数，它就能用 Duffie 和 Singleton 的方法根据观测到的价格和信用利差将其计量经济化地估计出来.[9] 此外，简化模型可以用来研究违约相关性. 这个模型的一个重要假设是，在已知违约强度样本路径的条件下，多个违约之间是相互独立的，这个假设也促进了被称作 Cox 过程的双重随机泊松过程的建立（Lando，1998).[10] 该假设意味着违约相关性和违约强度的相关性可以画上等号. 然而，Schonbucher 和 Schubert(2001) 指出，"通常，根据这个方法计算得到的违约相关性与实际经验中的违约相关性相比显得太小."

假设违约掉期的到期日为 T. 费用的溢价部分在时刻 $\widetilde{A}=\{T_1, T_2, \cdots, T_n\}$ 实现，总共有 n 个离散的支付日期；$\Delta_i=\Delta(T_{i-1}, T_i)$ 表示在 (T_{i-1}, T_i) 之间的天数部分. $T_0=0$ 在 0 时刻预先设定. 如果在到期日 T 之前发生违约，那么最大为违约日 τ；如果直到到期都没有发生违约，那么最大为到期日 T. 假设 $T_n\leqslant T$(通常 $T_n=T$). 令 $R\in(0, 1)$ 表示违约参考资产的回收率. 令 $\mathbf{1}_{\{\tau<T\}}$ 为指标函数，在 $\tau<T$ 时为 1 而在其余时间为 0. 令 $D(t, T)=B(t)/B(T)$ 表示银行账户计价，其中 $B(t)=B(0, t)=\exp\left(-\int_0^t r(u)\mathrm{d}u\right)$，而 $r(t)$ 表示即时的无风险利率，$t\in[T_{i-1}, T_i)$ ——也就是说，T_i 是 t 之后的第一个支付日. 此外令债券价格 $P(t, T)=E^Q\left(\frac{B(t)}{B(T)}\,|\,\Im_t\right)$，其中 $\Im_t$ 表示 t 时刻的信息过滤因子. 为了简化表示方法，用 $E_t[\cdot]$ 来表示 $E[\cdot\,|\,\Im_t]$. 令 E^Q 表示在风险中性概率测度下的期望值.

一般在估值中有三个假设条件：

1. 利率过程和违约过程在风险中性鞅测度 Q 下是相互独立的.
2. 违约只能在一组有限的、离散的日期发生.
3. 违约费用在违约发生的一刹那就已确定.

保护买方（PB）的总费用为：

$$\mathrm{CDS}^{PB}(t,T;\tau)=sN\sum_{i=1}^{n}D(t,T_i)\Delta_i\mathbf{1}_{\{\tau<T\}}+sND(t,\tau)\Delta_i\frac{(\tau-T_{i-1})}{(T_i-T_{i-1})}\mathbf{1}_{\{\tau<T_n\}} \tag{5.1}$$

其中最右边一项表示从之前的支付日到违约日 $\tau>t$ 之间，保护买方在违约时刻向保护卖方支付的累计利息 $A(T_{\beta(t)}, \tau)$. 因此，保护买方的费用可以被细分为从上一个支付日计起的溢价部分（PL）和累计溢价部分[11]（AP). 如果没有违约发生，那么保护卖方（PS）不向保护买方支付任何费用. 但是如果在到期日之前发生违约，那么卖方需要支付贷款中未被覆盖的部分，也就是：

$$CDS^{PS}(t,T;\tau)=(1-R)D(t,\tau)N\mathbf{1}_{\{\tau<T\}} \tag{5.2}$$

信用违约掉期在 t 时刻（$t>0$）的价值就是溢价部分（由保护买方支付）减去违约部分

（违约时由保护卖方支付）的价值.

$$CDS(t,T)=CDS^{PB}-CDS^{PS}$$

信用违约掉期的初始价值就是使得 CDS^{PB}（t，T；τ）和 CDS^{PS}（t，T；τ）在0时刻支付的费用相等的利差，在这种情况下，信用违约掉期的价值对于买卖双方来说都是零.令式（5.1）和式（5.2）相等可得：

$$sN\sum_{i=\beta(t)}^{n}D(0,T_i)\Delta_i\mathbf{1}_{\{\tau>T\}}+sND(0,\tau)\Delta_i\frac{(\tau-T_{\beta(t)-1})}{(T_i-T_{i-1})}\mathbf{1}_{\{\tau<T_n\}}=(1-R)D(0,\tau)N\mathbf{1}_{\{\tau<T\}}$$

于是有：

$$s=\frac{(1-R)D(0,\tau)\mathbf{1}_{\{\tau<T\}}}{\sum_{i=1}^{n}D(0,T_i)\Delta_i\mathbf{1}_{\{\tau>T_i\}}+D(0,\tau)\Delta_i\frac{(\tau-T_{\beta(t)-1})}{(T_i-T_{i-1})}\mathbf{1}_{\{\tau<T_n\}}}\tag{5.3}$$

或者

$$s=\frac{(1-R)\exp\left(-\int_0^{\tau}r(u)\mathrm{d}u\right)\mathbf{1}_{\{\tau<T\}}}{\sum_{i=1}^{n}\exp\left(-\int_0^{T_i}r(u)\mathrm{d}u\right)\Delta_i\mathbf{1}_{\{\tau>T_i\}}+\exp\left(-\int_0^{\tau}r(u)\mathrm{d}u\right)\Delta_i\frac{(\tau-T_{i-1})}{(T_i-T_{i-1})}}\tag{5.4}$$

因此，在时刻 $t(0<t<\tau)$ 信用违约掉期的净现值为：

$$CDS(t,T)=\mathbf{1}_{\{\tau>t\}}N\Big[s\sum_{i=1}^{n}D(t,T_i)\Delta_i\mathbf{1}_{\{\tau>T\}}+sD(t,\tau)\Delta_i\frac{(\tau-T_{i-1})}{(T_i-T_{i-1})}\mathbf{1}_{\{\tau<T_n\}}$$

$$-(1-R)D(t,\tau)\mathbf{1}_{\{\tau<T\}}\Big]\tag{5.5}$$

5.4 风险率函数

给出一个经过信息过滤的概率空间（Ω，$\Im$，$(\Im_t)_{t\geqslant0}$，P）令 τ 表示（$\Im_t$）——终止时刻，该时刻违约发生，违约的概率分布函数为 $F(t)=P(\tau\leqslant t)$，概率密度为 $f(t)$.令 P（$t<\tau<t+\Delta t|\tau>t$）表示违约发生在时间间隔（t，$t+\Delta t$）的概率，假定参考实体能够生存到时刻 t，定义风险率函数 $h=\lambda(t)$ 为：

$$\lambda(t)=\lim_{\Delta t\to0}P(t<\tau\leqslant t+\Delta t|\tau>t)$$

$\lambda(t)$、$f(t)$ 和 $F(t)$ 之间的关系可以通过如下方式建立：

$$\lambda(t)=\frac{P(t<\tau\leqslant t+\Delta t|\tau>t)}{P(\tau>t)}=\lim_{\Delta t\to0}\frac{\int_t^{t+\Delta t}f(u)\mathrm{d}u}{\int_t^{\infty}f(u)\mathrm{d}u}$$

$$=\frac{f(t)}{1-F(t)}=\frac{\frac{\partial}{\partial t}F(t)}{1-F(t)}=-\frac{\partial}{\partial t}\log(1-F(t))$$

解上述微分方程，可得：

$$F(t)=1-\exp\left(1-\int_0^{t}\lambda(u)\mathrm{d}u\right)\tag{5.6}$$

和

$$f(t) = \lambda(t)\exp\left(1 - \int_0^t \lambda(u)\mathrm{d}u\right) \tag{5.7}$$

从（5.5）中，定义存活率为 $S(t)=1-F(t)=P(\tau>t)$，那么可得：

$$S(t) = \exp\left(-\int_0^t \lambda(u)\mathrm{d}u\right) \tag{5.8}$$

需要注意的是，分布函数 $F(t)$ 和存活函数 $S(t)$ 给出了两个数学形式上相同的方法来表示直到违约时刻的随机变量的分布.[12]

5.5 泊松过程和 Cox 过程

为了计算公式（5.4）的值，我们需要给出违约时刻 τ（也就是信用事件发生造成的终止时刻）的模型. 我们认为违约时刻 τ 服从 Cox 过程. 如 Lando 所述，Cox 过程是一个广义的泊松过程，它的强度可以是随机的，因此已知一个特定的强度 $\lambda(\cdot,\omega)$，这个跳跃过程就成为一个强度为 $\lambda(s,\omega)$ 的非同质泊松过程.[13]在这样的过程下，τ 就表示时间不均匀的（双重随机泊松过程）计数（整数变量）过程 $N(t)$ 的第一个跳变时刻. 过程 $N(t)$ 定义如下：

$$N(t) := \begin{cases} \sum_{i\in N} \mathbf{1}_{\{\tau_i \leqslant t\}}, & t > 0 \\ 0, & t = 0 \end{cases}$$

其中，$\mathbf{1}_{\{\tau_i\leqslant t\}}$ 是（$\Im_t$）——终止时间 τ_i 的指标函数. 此外，该过程在 $0\leqslant t<T$ 满足以下条件：

1. $N(T)-N(t)$ 的增量独立于 σ——$\Im_t$.

2. $N(T)-N(t)$ 的增量的分布按照泊松定律，其变量是单调递增、非负并且可逆的（也称为风险函数）.

$$\Lambda(t) = \int_0^t \lambda(u)\mathrm{d}u$$

其中，$\lambda(t)$ 为一个非负的处处可测的强度过程[14]：

$$P(N(T) - N(t) = n) = \frac{\left(\int_t^T \lambda(u)\mathrm{d}u\right)^n}{n!}\exp\left(-\int_t^T \lambda(u)\mathrm{d}u\right).$$

特别地，$N(0)=0$，并且

$$P(N(0) = 0) = \exp\left(-\int_0^t \lambda(u)\mathrm{d}u\right)$$

并且 N 有一个静态独立增量. $\Lambda(\tau)$ 称为可预测补偿过程，它给出了很多关于下一时间的跳变概率的信息. $\Lambda(t)$ 是个增函数，这是由于 $N(t)$ 也是递增的. 该模型假定 $N(t+\Delta t)-N(t)$ 只能是 0 或 1，因此，

$$\begin{aligned} &E[\Lambda(t+\Delta t) - \Lambda(t) \mid \Im_t] \\ &= 1\cdot P[N(t+\Delta t) - N(t) = 1 \mid \Im_t] + 0\cdot P[N(t+\Delta t) - N(t) = 0 \mid \Im_t] \\ &= P[N(t+\Delta t) - N(t) = 1 \mid \Im_t] \end{aligned}$$

因此，补偿过程 $N(t)-\Lambda(t)$ 是一个局部鞅. 由于补偿因子是可以预测的，因此它在下一个时间段内的增量是确定的. 所以，补偿因子为计数过程的局部跳变概率提供了一个即时更新

的测度.[15]

时间内齐次过程 N 可以写成 $N(t)=M(\Lambda^{-1}(t))$，其强度恒为 1. M 是一个递增右连续的单位跳变，其静态增量为 $M(0)=0$. 如果 N 在 τ 时刻发生第一次跳变，那么 M 在 $\Lambda(\tau)$ 时刻发生第一次跳变. 因为 M 是一个强度为 1 的泊松过程，所以它在 $\Lambda(\tau)$ 发生第一次跳变是以参数为 1 的指数随机变量分布的，因此在风险中性概率测度 Q 下有：

$$Q(\Lambda(\tau)<s)=1-\exp(-s).$$

为了模拟出 N 的第一个跳变 τ，令 Y 是一个单位指数随机变量，而违约时间定义为：

$$\tau=\inf\left(t:\int_0^t\lambda(u)\mathrm{d}u\geqslant Y\right) \tag{5.9}$$

可以将强度过程写成一个当前状态变量级的函数. 这些状态变量包括时间、股票价格、信用评级以及其他一些预测违约可能性的相关变量. 在这种情况下，我们将强度表示为 $\lambda(X_t)$，其中 X 是一个 $\mathcal{R}^d$ 随机过程. 因此，为了简化，我们将风险率假设为一个只是关于时间的函数.

在一个风险中性概率测度 Q 下的风险中性的世界中，我们需要以下信息来进行设置（用 $\sigma(\cdot)$ 表示 σ 代数）：

$$\begin{aligned}G_t&=\sigma(\mathbf{1}_{\{\tau\leqslant u\}}:0\leqslant u\leqslant t)\\H_t&=\sigma(X_s:0\leqslant s\leqslant t)\\\Im_t&=G_t\vee H_t\end{aligned}$$

其中，我们假设存在概率空间 $\{\Omega,\Im,P\}$ 下的标准单位泊松过程 N 且其信息过滤因子为$\Im_t$，这个因子取决于直到时刻 t 对状态变量的了解程度以及直到该时刻是否有违约发生.[16]利用以下事实：

$$E[\mathbf{1}_{\{\tau\geqslant T\}}\mid\Im_t]=\mathbf{1}_{\{\tau>t\}}\exp\left(-\int_t^T\lambda(s)\mathrm{d}s\right)$$

利用在集合 $\{\tau\leqslant t\}$ 上条件期望为 0 以及集合 $\{\tau>t\}$ 是 G_t 的原子的条件，可以得到以下证明：[17]

$$\begin{aligned}E[\mathbf{1}_{\{\tau\geqslant T\}}\mid\Im_t]&=\mathbf{1}_{\{\tau>t\}}E(1_{\{\tau\geqslant T\}}\mid\Im_t)\\&=\mathbf{1}_{\{\tau>t\}}\frac{P(\{\tau\geqslant T\}\cap\{\tau>t\}\mid\Im_t)}{P(\tau>t\mid\Im_t)}=\mathbf{1}_{\{\tau>t\}}\frac{P(\tau\geqslant T\mid\Im_t)}{P(\tau>t\mid\Im_t)}\\&=\mathbf{1}_{\{\tau>t\}}\frac{\exp\left(-\int_0^T\lambda(s)\mathrm{d}s\right)}{\exp\left(-\int_0^t\lambda(s)\mathrm{d}s\right)}=\mathbf{1}_{\{\tau>t\}}\exp\left(-\int_t^T\lambda(s)\mathrm{d}s\right).\end{aligned}$$

还可以假设，无论违约强度是否确定，随机贴现因子 $D(t,T)=\exp\left(-\int_t^T r(u)\mathrm{d}u\right)$都和违约时刻 τ 无关.

5.6 用确定性强度模型进行估值

可以使用确定性强度模型对信用违约掉期进行估值，取式（5.4）的风险中性期望可得：

$$
\begin{aligned}
&= E^Q\Big\{\mathbf{1}_{\{\tau>t\}}N\Big[s\sum_{i=1}^{n}D(t,T_i)\Delta_i\mathbf{1}_{\{\tau>T_i\}} \\
&\quad + sD(t,\tau)\Delta_i\frac{(\tau-T_{i-1})}{(T_i-T_{i-1})}1_{\{\tau<T_n\}}-(1-R)D(t,\tau)\mathbf{1}_{\{\tau\leqslant T\}}\Big]\Big|\Im_t\Big\} \\
&= \mathbf{1}_{\{\tau>t\}}NE^Q\Big[s\sum_{i=1}^{n}E^Q[D(t,T_i)\mathbf{1}_{\{\tau>T\}}]\Delta_i \\
&\quad + sE^Q[D(t,\tau)\Delta_i\mathbf{1}_{\{\tau<T_n\}}]\frac{(\tau-T_{i-1})}{(T_i-T_{i-1})} \\
&\quad -(1-R)E^Q[D(t,\tau)\mathbf{1}_{\{\tau<T\}}]\Big|\Im_t\Big] \\
&= \mathbf{1}_{\{\tau>t\}}NE^Q\Big[s\sum_{i=1}^{n}P(t,T_i)\mathbf{1}_{\{\tau>T_i\}}\Delta_i \\
&\quad + s\Delta_iP(t,\tau)\mathbf{1}_{\{\tau<T_n\}}\frac{(\tau-T_{i-1})}{(T_i-T_{i-1})}-(1-R)P(t,\tau)\mathbf{1}_{\{\tau>T\}}\Big]\Big|\Im_t\Big] \\
&= \mathbf{1}_{\{\tau>t\}}N\Big[s\sum_{i=1}^{n}P(t,T_i)\Delta_i\exp\Big(-\int_t^{T_i}\lambda(s)\mathrm{d}s\Big) \\
&\quad + s\Delta_i\int_t^{T_n}P(t,u)\frac{(W-T_{i-1})}{(T_i-T_{i-1})}\mathrm{d}Q\{\tau\leqslant u\,|\,\Im_t\} \\
&\quad -(1-R)\int_t^{T}P(t,u)\lambda(u)\exp\Big(-\int_t^{u}\lambda(s)\mathrm{d}s\Big)\mathrm{d}u\Big]
\end{aligned}
\tag{5.10}
$$

因此有：

$$
\begin{aligned}
CDS(t,\tau,T,s) &= \mathbf{1}_{\{\tau>t\}}N\Big(s\sum_{i=1}^{n}P(t,T_i)\Delta_i\exp\Big(-\int_t^{T_i}\lambda(s)\mathrm{d}s\Big) \\
&\quad + s\Delta_i\int_t^{T_n}P(t,u)\Big(\frac{(u-T_{i-1})}{(T_i-T_{i-1})}\Big)\lambda(u)\exp\Big(-\int_t^{u}\lambda(s)\mathrm{d}s\Big)\mathrm{d}u \\
&\quad -(1-R)\int_t^{T}P(t,u)\lambda(u)\exp\Big(-\int_t^{u}\lambda(s)\mathrm{d}s\Big)\mathrm{d}u\Big]
\end{aligned}
\tag{5.11}
$$

如果将式（5.11）设为0来求解利差，可得在 $t=0$ 时，

$$
s=\frac{(1-R)\int_0^{T_n}\lambda(u)P(0,u)\exp\Big(-\int_0^{u}\lambda(s)\mathrm{d}s\Big)\mathrm{d}u}{\Big(\sum_{i=1}^{n}\Big\{\Delta_iP(0,T_i)\exp\Big(-\int_0^{T_i}\lambda(s)\mathrm{d}s\Big)+\int_{T_{i-1}}^{T_i}\Delta_i\lambda(u)\Big(P(0,u)\exp\Big(-\int_0^{u}\lambda(s)\mathrm{d}s\Big)\frac{(u-T_{i-1})}{(T_i-T_{i-1})}\Big)\mathrm{d}u\Big\}\Big)}
\tag{5.12}
$$

需要注意式（5.12）的分母是利差（RPV01）移动一个基点时的风险溢价，称为风险性PV01. O' Kane 和 Turnbull 指出保护的回收费用的积分表达式（式（5.12）中的分子）计算起来非常之繁琐. 在并未减少太多精确性的情况下，他们简单地假设信用事件只能在每年的几个离散日期发生有限的 M 次.[18] 对于一个期限为 T_n 的违约掉期，最多可以发生 $M\times T_n$ 次，$m=1,\cdots,M\times T_n$. 于是有：

$$PL=(1-R)\sum_{m=1}^{M\times T_n}P(0,T_m).\left(\exp\left(-\int_0^{T_{m-1}}\lambda(s)\mathrm{d}s\right)-\exp\left(-\int_0^{T_m}\lambda(s)\mathrm{d}s\right)\right) \tag{5.13}$$

M 的值越小，所需的计算量就越小，但是精确度也会随之降低. O'Kane 和 Turnbull (2003) 指出累计溢价的表达式：

$$sN\sum_{i=1}^{n}\int_{T_{i-1}}^{T_i}\Delta_i\lambda(u)P(0,u)\left(\exp\left(-\int_0^u\lambda(s)\mathrm{d}s\right)\frac{(u-T_{i-1})}{(T_i-T_{i-1})}\right)\mathrm{d}u$$

这个式子比较复杂，很难直接对其进行计算，但其近似于：[19]

$$\frac{sN}{2}\sum_{i=1}^{n}\Delta_i\lambda(u)P(0,u)\left((Q(0,T_{n-1})-Q(0,T_n))\frac{(u-T_{i-1})}{(T_i-T_{i-1})}\right)\mathrm{d}u \tag{5.14}$$

需要注意的是，如果在两个溢价支付日之间没有违约发生，那么平均累计溢价是全额溢价的一半，这是由于费用是在溢价期末进行支付的，并且我们假设风险率是一个每段时间内恒定的变量. 全额溢价为：

$$s\times N\times RPV01$$

其中：

$$RPV01=\sum_{i=1}^{n}\Delta_iP(0,T_i)\left[Q(0,T_i)+\frac{1_{AP}}{2}(Q(0,T_{i-1})-Q(0,T_i))\right]$$

其中，当信用违约掉期合约明确了累计溢价时，$1_{AP}=1$，其他情况下为 0，[20] 而 $Q(T_v, T_n)$ 是从时刻 $T_v(v=0, \cdots, n-1)$ 起，直到时刻 T_n 的参考实体的无套利存活率. 假设风险率在一段时间内是定值，我们定义：

$$Q(T_v,T_n)=\begin{cases}\exp(-\lambda_{0,1}\tau),\text{若 } 0<\tau\leqslant 1\\ \exp(-\lambda_{0,1}-\lambda_{1,3}(\tau-1)),\text{若 } 1<\tau\leqslant 3\\ \exp(-\lambda_{0,1}-2\lambda_{1,3}-\lambda_{3,5}(\tau-3)),\text{若 } 3<\tau\leqslant 5\\ \exp(-\lambda_{0,1}-2\lambda_{1,3}-2\lambda_{3,5}-\lambda_{5,7}(\tau-5)),\text{若 } 5<\tau\leqslant 7\\ \exp(-\lambda_{0,1}-2\lambda_{1,3}-2\lambda_{3,5}-2\lambda_{5,7}-\lambda_{7,10}(\tau-7)),\text{若 } \tau\geqslant 7\end{cases}$$

其中 $\tau=T_n-T_v$ 且假设风险率在 10 年期限之外维持不变.

假设风险率函数是每段为恒值，我们可以用式 (5.12) 来计算一个 1 年违约掉期：

$$s=\frac{(1-R)\int_0^{T_n}\lambda_{0,1}P(0,u)\exp(-\lambda_{0,1}u)\mathrm{d}u}{\sum_{i=1}^{n}\left\{\Delta_iP(0,T_i)\exp(-\lambda_{0,1}T_i)+\int_{T\,i-1}^{T_i}\lambda_{0,1}\Delta_iP(0,u)\left(\exp(-\lambda_{0,1}u)\frac{(u-T_{i-1})}{(T_i-T_{i-1})}\right)\mathrm{d}u\right\}}$$

或者用 O'Kane 和 Turnbull 的近似法来简化计算：

$$s=\frac{(1-R)\sum_{m=1}^{12}P(0,T_m)(\exp(-\lambda_{0,1}T_{m-1})-\exp(-\lambda_{0,1}T_m))}{\frac{1}{2}\sum_{\substack{i=3\\ Step 3}}^{12}\Delta(T_{i-3},T)P(0,T_i)(\exp(-\lambda_{0,1}T_{i-3})-\exp(-\lambda_{0,1}T_i))}$$

假设每季度支付一次，以月份进行离散化 ($M=12$)，并且是累计溢价.

通常，市场把信用违约掉期的利差 s 设为 $s^{mid}(t, T)$（买卖价格的平均），使得这个价格在

t 时刻是合理的. s 的值使 $CDS(t, T; \tau, s^{mid}(t, T), N)=0$. 在市场上，信用违约掉期 t 时刻的报价是通过对一系列的到期日 $T=t+1y$ 直到 $T=t+10y$ 的"合理"利差 $s^{mid}(t, T)$ 的买卖询价方式来确定的.

可以证明 Nelson-Siegel 曲线近似能够用来自举存活函数，并产生远期内涵违约概率曲线. 从 $S_0=1$ 开始，然后有：

$$S_1=\frac{\frac{1}{2}(1-R)(Z_0+Z_1)-s_1\tau_1(Z_0+2Z_1)/6}{\left(\left(\frac{1}{2}(1-R)(Z_0+Z_1)-s_1\tau_1(Z_0+2Z_1)/6\right)+s_1\tau_1 Z_1\right)}$$

以及

$$S_n=\frac{(1-R)\left(\sum_{i=1}^{n-1}(Z_{i-1}+Z_i)(S_{i-1}-S_i)+\frac{1}{2}(Z_{n-1}+Z_n)S_{n-1}\right)-s_n\left(\sum_{i=1}^{n-1}\frac{1}{6}(Z_{i-1}+2Z_i)\tau_i(S_{i-1}-S_i)+Z_iS_i\tau_i-\frac{1}{6}(Z_{n-1}+2Z_n)\tau_nS_{n-1}\right)}{\left(\frac{1}{2}(Z_{n-1}+Z_n)(1-R)-s_n\tau_n(Z_{n-1}+2Z_n)/3\right)+s_n\tau_nZ_n}$$

限制 $0\leqslant S_n\leqslant S_{n-1}\leqslant 1$，其中

- s_n 是到期为 n 的面值利差；
- Z_i 是从 0 到 T_i 的无风险折现因子；
- S_i 是从 0 到 T_i 的无违约概率；
- τ_i 表示从 T_{i-1} 到 T_i 的累计期限；
- $(S_{i-1}-S_i)$ 表示从 T_{i-1} 到 T_i 之间发生信用事件的概率.

5.7 风险率函数校准

为了简便起见，许多实践人员假定风险率函数是确定的，即恒定不变，$\Lambda(T)=\lambda$，或者说是每段时间上确定：

$$\Lambda(T)=\begin{cases}\lambda_{0,1}, & T_0<t\leqslant T_1\\ \lambda_{1,2}, & T_1<t\leqslant T_2\\ \cdots & \\ \lambda_{n-1,n}, & T_{n-1}<t\leqslant T_n\end{cases}\qquad(5.15)$$

如图 5.2 所示.

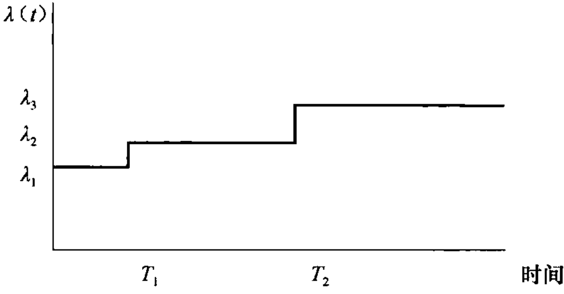


图 5.2 逐段线性风险函数

我们可以假设这条曲线在每段都是线性的. 然而，这只在以下情况会有所影响：(1) 对很多到期日，无法得知利差报价；(2) 该曲线陡度非常大. 这些情况通常不会出现，因为大部分合约只在作为 5 年期违约掉期时才具有流动性，因此可以假设曲线是水平的.[21] 主要的例外是对于反利差曲线，这往往和受困信用联系在一起.[22]

给出 1 年、3 年、5 年、7 年和 10 年的违约掉期利差并假设风险率函数的期限结构分为 5 个部分：$\lambda_{0,1}$、$\lambda_{1,3}$、$\lambda_{3,5}$、$\lambda_{5,7}$和 $\lambda_{7,10}$.

我们使用校准法中常用的假设，也就是假设风险率在信用违约掉期到期日间是每段恒定不变的. 如果用 $[T_1, T_2, \cdots, T_n]$ 表示市场上进行交易并有报价的信用违约掉期合约的到期时间（以年记），这就意味着风险函数 $\lambda(t)$ 可以表示为：

$$\lambda(t) = \sum_{i=1}^{n} c_i \mathbf{1}_{T_{i-1}, T_i}(t)$$

其中，c_i 为一系列正常数，$i=1, \cdots, n$.

这个假设意味着公式（5.4）中的分布函数 $F(t)$ 可以写成：

$$F(t) = 1 - \exp\left[-\sum_{j=1}^{k} c_j (T_j - T_{j-1})\right],$$
$$k = \begin{cases} 1, & t \leqslant t_1 \\ 2, & t_1 < t \leqslant t_2 \\ \cdots & \cdots \\ n, & t > t_{n-1} \end{cases} \tag{5.16}$$

将式（5.16）带入信用违约掉期利差定价方程（5.12），取最短到期日 T_1，并近似地为参考实体确定回收率，可以推导出 c_1 的值. 在确定了 c_1 之后，可以用相对于 T_2 到期的信用违约掉期利差来校准 c_2，然后依照这种方法校准所有剩余的 c_j，直到 T_n. 这种迭代求取风险率 c_j 的方法称为自举法(bootstrapping).

范例 1

假设在 2004 年 3 月 1 日（交易日），一个保护买方购买了一份针对福特公司名义价值为 1 000 万美元信用的三年期违约掉期保护，有效日应该在 $t+1$ 日，也就是 2004 年 3 月 2 日；交割日应该是$t+3$日，也就是 2004 年 3 月 7 日；而到期日应该在 2007 年 3 月 20 日，这是因为信用违约掉期在 3 月、6 月、9 月、12 月的 20 日到期. 通常，如果交割日或者支付日在周末或者节假日，违约掉期采用顺延一日记日法来计算. 例如，前两个支付日是 2004 年 3 月 22 日和 2004 年 6 月 21 日，因为 2004 年 3 月 20 日是周六而 2004 年 6 月 20 日是周日. 图 5.1 中三年期信用违约掉期的报价是 156 个基点.

表 5.1 列出了支付计划.

表 5.1

天数	期限（实际/360）	支付日期	支付金额（美元）	天数	期限（实际/360）	支付日期	支付金额（美元）
21	0.058 3	Mon. March 22，2004	9 094.80	92	0.255 6	Mon. March 21，2005	39 873.60
92	0.255 6	Mon. June 21，2004	39 873.60	93	0.258 3	Mon. June 21，2005	40 294.80
92	0.255 6	Mon. Sept 20，2004	39 873.60	92	0.255 6	Tue. Sept 20，2005	39 873.60
92	0.255 6	Mon. Dec 20，2004	39 873.60	92	0.255 6	Tue. Dec. 20，2005	39 873.60

（续）

天数	期限（实际/360）	支付日期	支付金额（美元）	天数	期限（实际/360）	支付日期	支付金额（美元）
91	0.252 8	Tue. March 20，2006	39 436.80	91	0.252 8	Wed. Dec. 20，2006	39 436.80
93	0.258 3	Tue. June 20，2006	40 294.80	91	0.252 8	Tues. Mar. 20，2007	39 436.80
93	0.258 3	Wed. Sept. 20，2006	40 294.80				

例如，图 5.3 所显示的屏幕是一组在 2004 年 3 月 5 日从 Bloomberg 截取的针对福特的信用违约掉期报价.

<HELP> for explanation.　　N201 Govt　　CDSD
1<GO> to save source and contributors, <Menu> to return

CONTRIBUTED CDS SPREADS

Enter the corporate ticker, currency and debt type to search again.
Corporate Ticker: F　　Currency: USD　　Debt Type: 3　　99) Search

1 Contributed Par CDS Spreads

	Term	Ticker	Contributor		Bid (bps)	Ask	Update Time	
2)	6 mo			*	82.50	97.50		
3)	1 yr	CFMCRU1	CBGN		82.50	97.50	16:19	
4)	2 yr			*	112.75	126.75		
5)	3 yr	CFMCRU3	CBGN		143.00	156.00	16:19	
6)	4 yr			*	162.58	172.82		
7)	5 yr	CFMCRU5	CBGN		182.00	189.50	16:20	Day Count: ACT/360
8)	7 yr			*	182.00	189.50		Frequency: Q
9)	10 yr	CFMCRU10		*	182.00	189.50		Recovery Rate: 0.40

* Piecewise linear, Flat extrapolation

Contributor Preferences: CBGN

Australia 61 2 9777 8600　Brazil 5511 3048 4500　Europe 44 20 7330 7500　Germany 49 69 920410
Hong Kong 852 2977 6000　Japan 81 3 3201 8900　Singapore 65 6212 1000　U.S. 1 212 318 2000
Copyright 2004 Bloomberg L.P.
G358-9-0 05-Mar-04 18:25:18

图 5.3　信用违约掉期利差

资料来源：Bloomberg.

注意，只有 1 年、3 年、5 年期给出了报价. 因为不同的经销商会给出不同的买卖报价，保护买家应该向多家经销商进行询问以获得最佳报价. 比如，买家可以从 Dresner Bank 这家经销商获得一个暗示报价，如图 5.4 所示.

由式（5.12），我们能够近似出福特公司的实际合理利差. 为此，我们要自举出国债收益曲线或者插值美国零息债券曲线. 从 Bloomberg 上找到 2004 年 3 月 5 日的国债和零息债券利率，如图 5.5 所示.

我们还可以从美国 LIBOR 互换曲线或者美国贴现票据曲线中找到贴现因子，如图 5.6 所示.[23]

我们用美国政府债券曲线来进行自举. 图 5.7 中的屏幕展示了美国政府零息债券和美国 LIBOR 互换（实际/360）. 两条曲线在 1～7 年的期限内非常接近. 在实务中，大部分交易台要用

美国 LIBOR 互换曲线或者欧洲 LIBOR 互换曲线来进行迭代自举运算.

N2N194 Govt DRCD

18:21 CDS AUTOS PAGE 1 / 1

Underlying		1 Year Bid	Ask	Time		3 Year Bid	Ask	Time		5 Year Bid	Ask	Time
DCX	1)	32	37	3/04	15)	65	70	3/04	29)	88	91	3/04
FORD MTR CRED	2)	95	105	3/04	16)	146	156	3/04	30)	186	191	3/04
G.M.A.C	3)	62	72	3/04	17)	118	128	3/04	31)	157	162	3/04
BMW	4)	18	23	3/04	18)	29	34	3/04	32)	38	41	3/04
CONTINENTAL	5)	16	22	3/04	19)	30	36	3/04	33)	44	50	3/04
PEUGEOT	6)	16	22	3/04	20)	36	42	3/04	34)	50	55	3/04
PORSCHE	7)	12	27	3/04	21)	28	40	3/04	35)	41	49	3/04
RENAULT	8)	16	22	3/04	22)	36	42	3/04	36)	50	55	3/04
SCANIA	9)				23)				37)			
VOLKSWAGEN	10)	28	33	3/04	24)	53	58	3/04	38)	65	70	3/04
MICHELIN (CF)	11)	21	26	3/04	25)	38	43	3/04	39)	52	57	3/04
VALEO	12)				26)				40)			
VOLVO	13)				27)				41)			
FIAT	14)				28)				42)			

ALL LEVELS ARE INDICATIVE. PLEASE CONTACT YOUR DrKW SALESPERSON FOR LIVE PRICES.

TRADING DESK +44 20 7475 3927

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2004 Bloomberg L.P.
G358-9-0 05-Mar-04 18:21:54

图 5.4 信用违约掉期汽车利差

资料来源：Bloomberg.

Page P089 Corp FMCH
Hit <PAGE> for more info or <MENU> for list of curves.

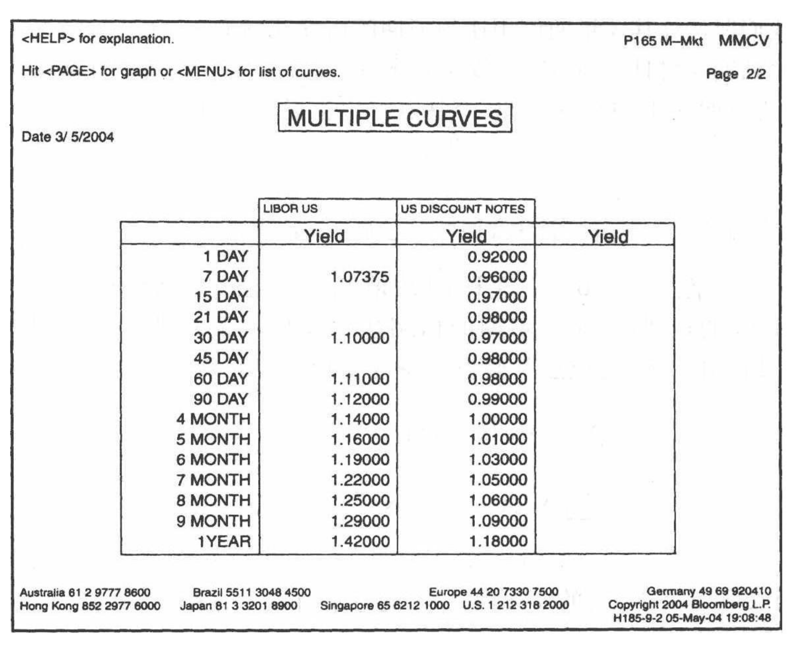
FAIR MARKET YIELD CURVES – HISTORY

Curve	90	80
Title	USD Govt	USD Trsy
Date	Agncy Zeros	Composite
3MO	3/ 5/04	3/ 5/04
6MO	1.2853	0.9202
1YR	1.3650	0.9559
2YR	1.3984	1.1034
3YR	1.9668	1.5751
4YR	2.4743	2.0784
5YR	2.9214	2.5121
7YR	3.3146	2.8465
8YR	3.8891	3.3884
9YR	4.0665	3.6600
10YR	4.3047	3.7455
15YR	4.5120	3.9580
20YR	5.0496	4.4816
25YR	5.3044	4.7609
30YR	5.3225	4.8929
	5.4648	4.6968

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2004 Bloomberg L.P.
H185-9-2 05-May-04 19:08:48

图 5.5 合理市场利益曲线

资料来源：Bloomberg.


<HELP> for explanation. P165 M–Mkt MMCV

Hit <PAGE> for graph or <MENU> for list of curves. Page 2/2

MULTIPLE CURVES

Date 3/ 5/2004

	LIBOR US	US DISCOUNT NOTES	
	Yield	Yield	Yield
1 DAY		0.92000	
7 DAY	1.07375	0.96000	
15 DAY		0.97000	
21 DAY		0.98000	
30 DAY	1.10000	0.97000	
45 DAY		0.98000	
60 DAY	1.11000	0.98000	
90 DAY	1.12000	0.99000	
4 MONTH	1.14000	1.00000	
5 MONTH	1.16000	1.01000	
6 MONTH	1.19000	1.03000	
7 MONTH	1.22000	1.05000	
8 MONTH	1.25000	1.06000	
9 MONTH	1.29000	1.09000	
1YEAR	1.42000	1.18000	

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2004 Bloomberg L.P.
H185-9-2 05-May-04 19:08:48

图 5.6 美国 LIBOR 互换曲线

资料来源：Bloomberg.

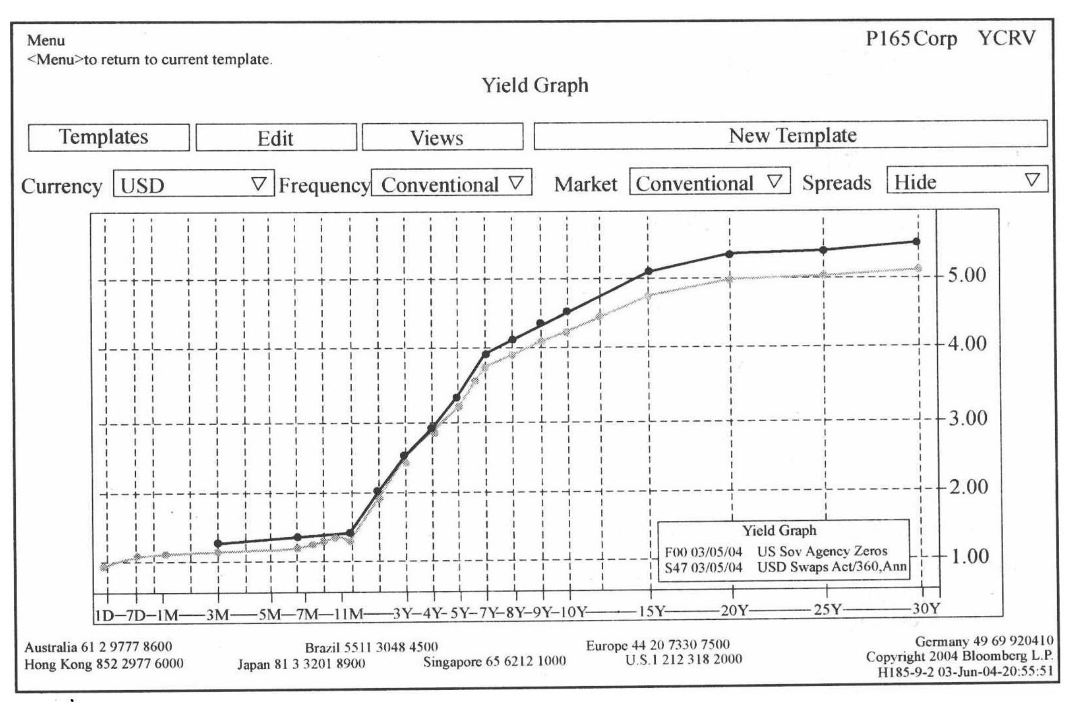


图 5.7 美国 LIBOR 和贴现票据利率

资料来源：Bloomberg.

我们可以消去式（5.12）中的内涵风险利率. 假设风险利率逐段为常数，即 $\lambda(S)=h$. 假设针对

福特公司信用，我们要计算其 3 年期信用违约掉期的内涵违约概率. 在这种情况下，$T=1$，并假设 $R=0.40$. 对所有的到期支付日，在自举出收益曲线后，我们可以计算出所有贴现债券的价值.

一年信用违约掉期的平均利率是 90 个基点，也就是 0.009. 可以将式（5.12）改写成：

$$s=\frac{(1-R)\int_0^1 hP(0,u)\exp(-\int_0^u h\mathrm{d}s)\mathrm{d}u}{\left(\sum_{i=1}^{n}\left\{\Delta_i P(0,T_i)\exp\left(-\int_0^{T_i}h\mathrm{d}s\right)+\int_{T_{i-1}}^{T_i}\Delta_i hP(0,u)\left(\exp\left(-\int_0^u h\mathrm{d}s\right)\frac{(u-T_{i-1})}{(T_i-T_{i-1})}\right)\mathrm{d}u\right\}\right)} \tag{5.17}$$

这个积分可以用像 Simpson 法这样的数值积分方法来求解. 但更实用的方法是使用 O'Kane和 Turnbull 的近似法. 对一年期的信用违约掉期（最初的 5 个支付日如图 5.4 所示）使用 O'Kane 和 Turnbull 的近似法并假设每季度支付，以月份进行离散化（$M=12$），可得：

$$0.009=\frac{0.6\sum_{m=1}^{12}P(0,T_m)(\exp(-\lambda_{0,1}T_{m-1})-\exp(-\lambda_{0,1}T_m))}{(1/2)\sum_{i=1}^{5}\Delta_i P(0,T_i)(\exp(-\lambda_{0,1}T_{i-1})+\exp(-\lambda_{0,1}T_i))}$$

其中 $T_0=0.0$，$T_1=0.0833$，$T_2=0.167$，$T_3=0.25$，…，$T_{12}=1.00$. 求解 $\lambda_{0,1}$ 必须用到数值迭代方法，这是因为上述方程是关于 $\lambda_{0,1}$ 的非线性方程. 通过求解以上方程[24]得到 $\lambda_{0,1}=0.016338=1.63\%$. 为了计算 $\lambda_{1,3}$，求解以下方程：

$$0.01495=\frac{0.6\sum_{m=1}^{36}P(0,T_m)(\exp(-\lambda_{0,1}-\lambda_{1,3}(T_{m-1}-1))-\exp(-\lambda_{0,1}-\lambda_{1,3}(T_m-1)))}{(1/2)\sum_{i=1}^{13}\Delta_i P(0,T_i)(\exp(-\lambda_{0,1}-\lambda_{1,3}(T_{i-1}-1))+\exp(-\lambda_{0,1}-\lambda_{1,3}(T_i-1)))}$$

求得 $\lambda_{1,3}=0.02739=2.74\%$.

要计算 $\lambda_{3,5}$，则需求解：

$$0.01885=\frac{\left(0.6\sum_{m=1}^{60}P(0,T_m)(\exp(-\lambda_{0,1}-\lambda_{1,3}(T_{m-1}-1)-\lambda_{3,5}(T_{m-1}-3))-\exp(-\lambda_{0,1}-\lambda_{1,3}(T_m-1)-\lambda_{3,5}(T_m-3)))\right)}{\left((1/2)\sum_{i=1}^{21}\Delta_i P(0,T_i)(\exp(-\lambda_{0,1}-\lambda_{1,3}(T_{i-1}-1)-\lambda_{3,5}(T_{i-1}-3))+\exp(-\lambda_{0,1}-\lambda_{1,3}(T_i-1)-\lambda_{3,5}(T_i-3)))\right)}$$

求得 $\lambda_{3,5}=3.69\%$.

为了计算风险率，这里使用了一个 Hazard 类（风险类）来包括所涉及的计算：

HazardRate.h

```
#ifndef _HAZARD__
#define _HAZARD__

#include <map>
#include <vector>
#include <math.h>

#define R 0.6  // recovery rate
#define absolute(x) (x > 0) ? x : -x
```

```
#define MAX_ITER    25
#define THIRTY_SIX  36
#define TWELVE      12
#define SIXTY       60
#define EIGHTY_FOUR 84

class HazardRate
{
  public:
        HazardRate() {}
        HazardRate(std::vector<double> sp) : spread(sp) {}
        virtual ~HazardRate() {}
        void init();       // initialize and store tenors and
                           // discount rates
        void calcDiscountBonds();  // calculates discount bond
                                   // prices
        double calcHazard(int num);  // calculates hazard rates
        double calcDeriv(double h, int num);   // calculates first
                                               // derivatives in
        double calcFunc(double h, int num);
        double Q(int num, double h, double t);
  private:
        std::vector<double> P;  // stores discount coupon bond
                                // prices
        std::map<int,double> T;  // stores zero coupon maturities
        std::vector<double> delta; // stores tenors between payment
                                   // dates
        std::vector<double> r;   // stores discount rates
        std::vector<double> haz;   // stores hazard rates
        std::vector<double> spread;  // stores spread quotes
};

#endif _HAZARD__
```

首先我们需要读入并计算所有期限和贴现率.期限可以通过信贷还款计划来计算，而贴现率可以从收益曲线中自举出来.数据既可以来源于文本文件也可以通过 API 编程来源于一些数据源.[25]假设利差数据如下：

1 年	82.50	97.50
3 年	143.00	156.00
5 年	182.00	189.50
7 年	195.00	200.50

在主程序中从文本文件中读入利差数据：

Hazard_Main.cpp

```
#include <strstrea.h>
#include <fstream.h>
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include "Hazard.h"

void main()
{

  std::vector<double> mat;
  char buffer[SIZE_X];
  char dataBuffer[SIZE_X];
  std::vector<double> spread;
  char* str = NULL;
  double yr = 0.0;
```

```
    double bid = 0.0;
    double ask = 0.0;
    // path to file could be read in as an argument to main function
    const char* file = "c:\\spreads.txt";  // file name

    ifstream fin;  // input file stream
    fin.clear();
    fin.open(file);

    if (fin.good())
    {
       while (!fin.eof())
       {
          fin.getline(buffer,sizeof(buffer)/sizeof(buffer[0]));
          istrstream str(buffer);
          // Get data
          str >> dataBuffer;
          yr = atof(dataBuffer);

          str >> dataBuffer;
          if (strcmp(dataBuffer,"MO") == 0)
            yr = (double) yr/12;
          else if (strcmp(dataBuffer,"WK") == 0)
            yr = (double) yr/52;
          else if (strcmp(dataBuffer,"DY") == 0)
            yr = (double) yr/365;

          mat.push_back(yr);
          str >> dataBuffer;
          bid = atof(dataBuffer);

          str >> dataBuffer;
          ask = atof(dataBuffer);
          spread.push_back((double)((bid + ask)/2)/10000);
       }
    }
    else
      cout << "File not good!" << "\n";

    fin.close();

}
```

用来计算这些数据的 C++代码如下：

HazardRate.cpp

```
#include "HazardRate.h"

void HazardRate::calcDiscountBonds()
{
        for (int i = 0; i <= EIGHTY_FOUR; i++)
                P.push_back(exp(-r[i]*T[i]));
}

double HazardRate::calcHazard(int num)
{
        double h = 0.01;
        double temp = h;
        double diff = 0.0;
```

```
        double f, f1 = 0.0;
        const double error = 0.000001;
        int cnt = 0;

        do
        {
                temp = h;
                f = calcFunc(h,num);
                f1 = calcDeriv(h,num);
                h = h - f/f1;
                diff = h - temp;
                cnt++;

        }
        while (absolute(diff) > error);
        haz.push_back(h);
        return h;
}

double HazardRate::calcFunc(double h,int num)
{

        double sum = 0;
        double sum1 = 0;
        double v = 0.0;

        if ((num > 0) && (num <= 1))
        {
                for (int i = 1; i <= 5; i++)
                        sum = sum + delta[i-1]*P[i-1]*(Q(num,h,T[i-1]) +
                          Q(num,h,T[i]));

                for (i = 1; i <= 12; i++)
                        sum1 = sum1 + P[i-1]*(Q(num,h,T[i-1]) -
                          Q(num,h,T[i]));

                v = R*sum1 - 0.5*spread[0]*sum;
        }
        else if ((num > 1) && (num <= 3))
        {
                for (int i = 1; i <= 13; i++)
                        sum = sum + delta[i-1]*P[i-1]*(Q(num,h,T[i-1]) +
                          Q(num,h,T[i]));

                for (i = 1; i <= 36; i++)
                        sum1 = sum1 + P[i-1]*(Q(num,h,T[i-1]) -
                          Q(num,h,T[i]));

                v = R*sum1 - 0.5*spread[1]*sum;

        }
        else if ((num > 3) && (num <= 5))
        {
                for (int i = 1; i <= 21; i++)
                        sum = sum + delta[i-1]*P[i-1]*(Q(num,h,T[i-1]) +
                          Q(num,h,T[i]));

                for (i = 1; i <= 60; i++)
                        sum1 = sum1 + P[i-1]*(Q(num,h,T[i-1]) -
                          Q(num,h,T[i]));
```

```
                v = R*sum1 - 0.5*spread[2]*sum;

        }
        else if ((num > 5) && (num <= 7))
        {
                for (int i = 1; i <= 29; i++)
                        sum = sum + delta[i-1]*P[i-1]*(Q(num,h,T[i-1]) +
                          Q(num,h,T[i]));

                for (i = 1; i <= 84; i++)
                        sum1 = sum1 + P[i-1]*(Q(num,h,T[i-1]) -
                          Q(num,h,T[i]));

                v = R*sum1 - 0.5*spread[3]*sum;
        }
        else if (num > 7)
        {
                for (int i = 1; i <= 41; i++)
                        sum = sum + delta[i-1]*P[i-1]*(Q(num,h,T[i-1]) +
                          Q(num,h,T[i]));

                for (i = 1; i <= 120; i++)
                        sum1 = sum1 + P[i-1]*(Q(num,h,T[i-1]) -
                          Q(num,h,T[i]));

                v = R*sum1 - 0.5*spread[4]*sum;
        }

        return v;
}

double HazardRate::calcDeriv(double h, int num)
{
        double sum = 0;
        double sum1 = 0;
        double v = 0.0;
        int i = 0;

        if ((num > 0) && (num <= 1))
        {
          for (i = 1; i <= 5; i++)
                sum = sum + delta[i-1]*P[i-1]*(T[i-1]*Q(num,h,T[i-1]) +
                  T[i]*Q(num,h,T[i]));

           for (i = 1; i <= 12; i++)
                sum1 = sum1 + P[i-1]*(-T[i-1]*Q(num,h,T[i-1]) +
                  T[i]*Q(num,h,T[i]));

            v = R*sum1 - 0.5*spread[0]*sum;

        }
        else if ((num > 1) && (num <= 3))
        {
            for (i = 1; i <= 13; i++)
              sum = sum + delta[i-1]*P[i-1]*((T[i-1]-1)*Q(num,h,T[i-1]) +
                        (T[i]-1)*Q(num,h,T[i]));

            for (i = 1; i <= 36; i++)
               sum1 = sum1 + P[i-1]*(-(T[i-1]-1)*Q(num,h,T[i-1]) +
```

```
                (T[i]-1)*Q(num,h,T[i]));

        v = R*sum1 - 0.5*spread[1]*sum;
    }
    else if ((num > 3) && (num <= 5))
    {
        for (i = 1; i <= 21; i++)
          sum = sum + delta[i-1]*P[i-1]*((T[i-1]-1)*Q(num,h,T[i-1]) +
          (T[i]-1)*Q(num,h,T[i]));

        for (i = 1; i <= 60; i++)
            sum1 = sum1 + P[i-1]*(-(T[i-1]-3)*Q(num,h,T[i-1]) +
              (T[i]-1)*Q(num,h,T[i]));

        v = R*sum1 - 0.5*spread[2]*sum;

    }
    else if ((num > 5) && (num <= 7))
    {
        for (i = 1; i <= 29; i++)
          sum = sum + delta[i-1]*P[i-1]*((T[i-1]-1)*Q(num,h,T[i-1]) +
          (T[i]-1)*Q(num,h,T[i]));

       for (i = 1; i <= 84; i++)
         sum1 = sum1 + P[i-1]*(-(T[i-1]-5)*Q(num,h,T[i-1]) +
           (T[i]-1)*Q(num,h,T[i]));

        v = R*sum1 - 0.5*spread[3]*sum;
    }
    else if (num > 7)
    {
       for (int i = 1; i <= 41; i++)
            sum = sum + delta[i-1]*P[i-1]*(Q(num,h,T[i-1]) +
              Q(num,h,T[i]));

       for (i = 1; i <= 120; i++)
          sum1 = sum1 + P[i-1]*(-(T[i-1]-7)*Q(num,h,T[i-1]) +
            (T[i]-1)*Q(num,h,T[i]));

       v = R*sum1 - 0.5*spread[4]*sum;
     }

     return v;

}

double HazardRate::Q(int num, double h, double T)
{

     double q = 0.0;

     if ((num > 0) && (num <= 1))
             q = exp(-h*T);
     else if ((num > 1) && (num <= 3))
             q = exp(-haz[0] - h*(T-1));
     else if ((num > 3) && (num <= 5))
             q = exp(-haz[0] - 2*haz[1] - h*(T-3));
     else if ((num > 5) && (num <= 7))

             q = exp(-haz[0] - 2*haz[1] - 2*haz[2] - h*(T-5));
```

```
        else // assume hazard rate remains constant beyond 10Y maturity
                q = exp(-haz[0] - 2*haz[1] - 2*haz[2] - 2*haz[3] - h*(T-7));

        return q;
}
```

期限和贴现率由用户输入或者用一个初始化程序来从文件读入.

对收益曲线进行自举并对福特公司的信用市场报价风险率进行校准后，对于不同回收率假设，可以画出如图 5.8 所示的违约时间分布函数曲面.

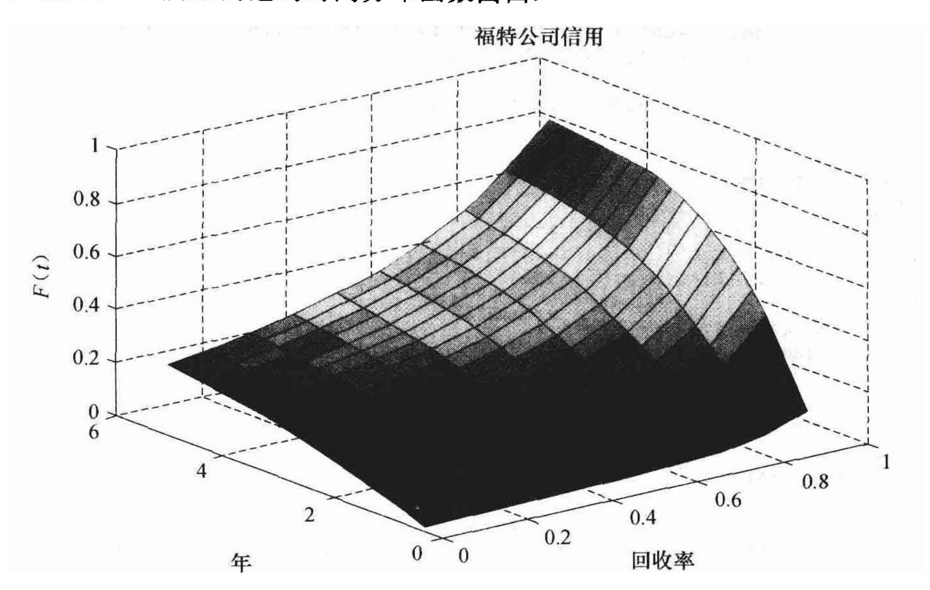


图 5.8 违约时间分布函数曲面

资料来源：Galiani S(2003)，pg. 36.

5.8 信用曲线的建立和校准

如先前所讨论的，为了对信用衍生品进行定价，需要通过像债券或资产互换利差这样流动性强的衍生品价格来构建信用曲线. 构建过程中，可以利用 Duffie 和 Singleton（1997）中的为可违约息票债券定价的简化模型. 通过对可违约债券所许诺的现金流用一个经信用风险调整的贴现因子进行贴现，可以计算出该债券的价值. 大致来说，如果造成违约的内在因素和影响利率的因素是彼此独立的，信用风险调整贴现因子（或总贴现因子）就是无风险贴现因子和纯信用风险贴现因子的乘积.[26]因此，债券的价值就是：

$$V(0)=\sum_{i=1}^{n}C_i\exp\left(-\int_0^{t_i}(r(s)+(1-R)h(s))\mathrm{d}s\right)$$

其中 C_i 是在时刻 t_i 所许诺的现金流. 和通常的假设一样，假设内涵远期违约风险率逐段为常量，即 $h(t)=h_i$，而内涵回收率为 $R=R_i$，$t_{i-1}<t<t_i$，有：

$$V(0)=\sum_{i=1}^{n}C_i\exp\left(-\int_0^{t_i}(r(s)+(1-R_i)h_i)\mathrm{d}s\right)$$

总的来说，用以下步骤来构建信用曲线：

1. 用 LIBOR 互换曲线的迭代程序自举出远期利率曲线.
2. 用校准/估计的方法计算出期望回收率.
3. 对利差曲线进行校准从而求得内涵风险率.

有时，如果假设每个债务人的回收率是根据类似信用评级的债务人的历史违约率和回收率来确定的，可以跳过步骤 2. 此外，对违约风险率期限结构的校准（步骤 3）和对期望回收率的校准（步骤 2）是分开的，这是由于联合校准通常会造成不稳定的结果.[27]实际情况中，回收率的确定受时间延迟、经销商询价和交割选项的影响.

在步骤 3 中，风险率校准法假定了一个参数表，其中确定了函数形式 $h(0, s)$ 并且对从所求价格中获取的对市场价格离散程度的测量指标（如均值方差）进行了最小化处理. 另一种方法是采用自举法，首先假定风险率是每段为常量，然后从较短的到期日开始，用和自举利率期限价格相似的方法对风险率进行自举.

5.9 一揽子信用违约掉期定价

贷款和债券组合经理人往往会发现，购买针对一揽子信用的（多名）违约掉期会比分别购买针对每一个贷款人的（单名）信用违约保护更便宜. 像基于 copula 函数的框架这样的一揽子违约掉期的定价法，能够捕捉到这些贷款人之间的相互关联，从而在篮子的合理定价中反映出来. 篮子中信用的违约相关性越高，篮子的价格就越低. 这是因为不同信用表现相似的情况下，整个篮子就表现得和一个单独的信用相类似. 比如，一揽子包含所有汽车生产商（如福特、通用、丰田、戴姆勒一克莱斯勒）的信用会比一个包含了多个行业的信用篮子更便宜.

篮子中的汽车行业信用都会受到汽车市场和经济形势的系统违约风险，因此，如果信用间违约相关性很高，并且信用结构相互依赖，那么一笔信用的违约概率的增加往往伴随着其他信用违约概率的增加.[28]也就是说，一笔信用的违约（违约终止时刻）会伴随着其他信用的违约（其他违约终止时刻). 反之，如果一揽子包含了一组违约相关性较低的多样化的信用，那么它的价格就近似于篮子是每一个信用保护的和，因此也就昂贵多了. 如果信用间的违约相关性低，那么一笔信用的违约就可能不会伴随着另一笔信用的违约.

可以用融入了 copula 方法的广义蒙特卡罗模拟来为一揽子信用进行定价. 对其中多笔信用的违约时刻的模拟是其中的一个重要步骤.

5.9.1 相关违约终止时刻的产生

在信用组合中，违约时刻的模拟与 copula 函数的选取有着错综复杂的联系，这个 copula 函数用于捕获贷款人之间的依赖性结构，并确定用于对每个实体定义存活函数的单变量边际分布函数的参数形式. 基于这点，Li 提出将这个模拟算法分两步实现：

1. 从 N 维 copula 生成 $[0, 1]$ 上相关的一致随机变量.
2. 通过边际分布函数的逆将相关均匀随机变量变换为违约时刻.

第一步是指定 copula 函数的特殊选择；第二步与 5.5 节中讨论的违约时刻到达设置有关.

5.9.2 从椭圆 copulae 抽样

多元高斯 copula

从高斯 copula C_R^{Gaussian}（其中 $\boldsymbol{R}$ 为相关矩阵）生成随机变量的过程如下：

1. 找到 $\boldsymbol{R}$ 的一个合适分解（如 Cholesky 分解）矩阵 $\boldsymbol{A}$，使得 $\boldsymbol{R}=\boldsymbol{A}\cdot\boldsymbol{A}^{\mathrm{T}}$.
2. 作一个不相关的标准正态变量的 N 维向量 $\boldsymbol{z}=(z_1, z_2, \cdots, z_N)$.
3. 令 $\boldsymbol{x}$：$\boldsymbol{z}'\boldsymbol{A}$.
4. 通过计算 $\boldsymbol{u}=\Phi(\boldsymbol{x})$，使 $\boldsymbol{x}$ 映射到 $[0, 1]$ 上均匀变量的 N 维随机向量 $\boldsymbol{u}$.

那么 $u\sim C_R^{\mathrm{Gaussian}}$.

多元学生 t copula

从高斯 copula $C_{R,v}^{\mathrm{Student}}$ 生成随机变量的过程如下，其中 $\boldsymbol{R}$ 为相关矩阵，v 为自由度：

1. 找到 $\boldsymbol{R}$ 的一个合适分解（如 Cholesky 分解）矩阵 $\boldsymbol{A}$，使得 $\boldsymbol{R}=\boldsymbol{A}\cdot\boldsymbol{A}^{\mathrm{T}}$.
2. 作一个不相关的标准正态变量的 N 维向量 $\boldsymbol{z}=(z_1, z_2, \cdots, z_N)'$.
3. 作一个独立的 χ_v^2 随机变量 s.
4. 令 $\boldsymbol{y}:=\boldsymbol{z}'\boldsymbol{A}$.
5. 令 $\boldsymbol{x}:=\boldsymbol{y}\sqrt{\frac{v}{s}}$.
6. 通过计算 $\boldsymbol{u}=t_v(\boldsymbol{x})$，将 $\boldsymbol{x}$ 映射到 $[0, 1]$ 上均匀变量的 N 维随机向量 $\boldsymbol{u}$.

那么 $\boldsymbol{u}\sim C_{R,v}^{\mathrm{Student}}$.

图 5.9 显示了来自高斯 copula 和学生 t copula 的 3 000 个样本. 注意与高斯 copula 相比，学生 t copula 在右上侧区域和左下侧区域累积的样本数，因此符合在极端事件出现时建模中尾部相关性的影响.

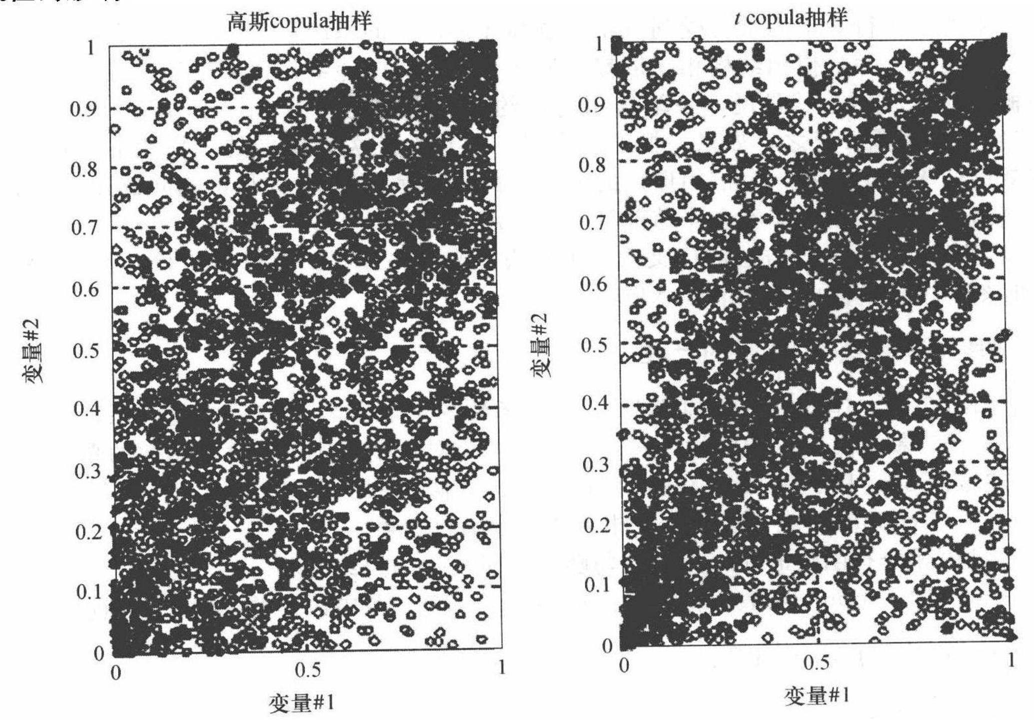


图 5.9　来自高斯 copula 和学生 t copula 的 3 000 个样本

资料来源：Galiani S(2003)，pg. 36

Matlab 实现如下：

copula_sample.m

```
function M=copula_sample(n,r)

% this function plots random variables extracted by a bivariate gaussian
% and Student's t copula with correlation coefficient "r" and "n" degrees
% of freedom

corr=ones(2);
corr(1,2)=r;
corr(2,1)=r;
% Now we use the simulation algorith suggested by ELM(01), pag. 26

A=chol(corr);
z=randn(2,1);
s=chi2rnd(n);
y=(z'*A)';
Gauss_U=normcdf(y);
x=(sqrt(n)/sqrt(s))*y;
t_U=tcdf(x,n);
M=[Gauss_U',t_U'];
```

copula_comp.m

```
function sample=copula_comp(num,n,r)
% this function shows bivariate r.v.s. from a normal and t copula with
% "num" simulation runs, "n" degrees of freedom and correlation "r"

U=zeros(num,4);
for k=1:num
    U(k,:)=copula_sample(n,r);
end
subplot(1,2,1); plot(U(:,1),U(:,2),'ob','MarkerSize',2)
grid on;
title('Gaussian copula Sampling');
xlabel('Variate #1');
ylabel('Variate #2');
subplot(1,2,2); plot(U(:,3),U(:,4),'or','MarkerSize',2)
grid on;
title('t copula Sampling');
xlabel('Variate #1');
ylabel('Variate #2');
sample=U;
```

5.9.3 违约到达时刻的分布

Schonbucher 注意到，对于每个运行的模拟，违约集中的路径是已知的，为了估计 N 个贷款人的违约分布，我们采用下面的模拟过程：

1. 对来自 copula C 的均匀变量的 N 维向量 $\boldsymbol{u}=(u_1, u_2, \cdots, u_N)$ 进行模拟.

2. 对每个实体 $n\in 1, 2, \cdots, N$，使用 5.7 节讨论的方法校准方程（5.10）中的存活函数 $\lambda_n(t)$.

5.9.4 一揽子 CDS 定价算法

下面给出计算一揽子 CDS 价值所必需的步骤。它可以用于像信用指数这样的定价工具，如 TracX、Iboxx 和 Dow Jones Credit Index.

1. 输入开始日、到期日、支付频数（如按季度支付）和 CDS 的名义金额.

2. 计算每个开始日和支付结束日之间的天数，习惯使用实际天数/360 天.[29]

3. 生成支付日期计划表.

4. 从诸如美国互换曲线（如 Bloomberg 上的 1YC1——可以使用 Bloomberg API 来提取报价或将它们输入到 Excel）的数据源获得期限结构，如互换曲线.

5. 自举互换曲线以生成对应于支付日的贴现因子.

6. 对篮子中的每个参考实体，校准市场交易 CDS 价差曲线. 因此，对每个参考实体，通过计算 $\lambda_n(t)$（风险利率）来校准违约时间分布函数的贴现因子，其中 $t=[T_1, T_2, \cdots, T_M]$，市场中可获得的 CDS 到期日的集合，使得每个这个名字的风险利率复制市场交易 CDS 报价.[30]

7. 计算步骤 6 中校准风险利率的存活和违约概率.

8. 对为建模篮子中贷款人之间的依赖性而选择的 copula 函数的参数进行校准.

9. 生成参考名字的相关矩阵 $\boldsymbol{R}$，并进行 Cholesky 分解以生成模拟违约时刻的相关偏差，例如，找到下三角矩阵 $\boldsymbol{A}$，使得 $C=\boldsymbol{A}\boldsymbol{A}'$.

10. 对每个模拟，重复下面的步骤：

(a) 使用 5.9.2 节介绍的方法生成相关均匀随机变量的 N 维向量.

(b) 对每个贷款人，使用 5.9.3 节介绍的方法将相应的均匀变量变换为违约时刻.

(c) 按降序排列违约时刻 τ 的 N 维向量，并根据违约的优先级（第 k 次违约），选择第 k 个坐标.

(d) 基于顺序统计 $\tau_{(k)}$ 的具体实现，计算溢价支付的贴现价值、累计利息和违约支付金额.

11. 计算前面数量的算术平均并应用方程（5.12）确定公平价差.

步骤 10 和步骤 11 通过以下伪代码给出.

```
Step 10:

      // initialize variables
      inflows = 0
      outflows = 0
      M = number of simulations

      // initialize random number generator (RNG), e.g. Mersenne
      // Twister
     for i = 1 to M
     {

         for each reference name in the basket
         {
```

```
// generate default times
generate uniform random number U from RNG for each name in
the basket and store in vector

generate independent normal variate z corresponding to each
U, e.g. z = Φ^-1(U).

generate correlated normal deviates from Cholesky matrix R,
R = AA', with independent normal deviates, e.g. z'A

if (correlated deviate < default probability)
  identify and store identity of which name defaulted

if CDS curve is flat
  default time = ln(1-correlated deviate)/hazard rate
else       // in reality CDS term structure is not flat
  default time = solved numerically    // see Li paper( )

Store default times
}
Sort the N-dimensional vector of default times τ in ascending
order, and according to the seniority of the contract (k-th
to default), select the k-th coordinate.
if min default time > maturity of the CDS basket
// no defaults occurred
{
   inflow = 0               // buyer protection receives nothing
   total_protection_leg = sum of notional*(actual/360)*(spread)
   outflow = outflow + total_protection_leg
}
else
{
  // use recovery rate of identified defaulted obligor
  inflow = inflow + notional*(1-recovery rate)*(discount rate
                     corresponding to default time)

  // must take into account interest accrued on default times
  // that do not fall exactly on a payment date. When you
  // default mid-period, the protection buyer is only liable
  // for the portion of protection payment on a pro-rata
  // basis

  sum_payments = sum of discounted protection payments up to
                 time of default
        // =(value of 1 basis point)*(min
        // default time of all times in the basket)
  outflow = outflow + sum_payments
}
}
```

Step 11:

```
basket payoff = average inflow -- average outflow
basket spread = (average inflow/average outflow)*(spread)
```

5.10 Matlab 中的信用篮子定价

interpolated_zero.m

```
function rate=interpolated_zero(t,zcr)
% This function computes the zero rate for a specific date with linear
% interpolation

for j=1:16                    % loop through all the zero rates date knots
    if zcr(j,1)>t & t>(1/12);
        rate=zcr(j-1,2)+((t-zcr(j-1,1))/(zcr(j,1)-zcr(j-1,1)))*
          ((zcr(j,2)-zcr(j-1,2)));
    elseif t<(1/12)
        rate=zcr(j,2);
        break
    end

end
```

premium_leg.m

```
function PV=premium_leg(expiry,def_time,disc_fact)

% This function calculate the present value of the premium leg of a basket
% default swap. We can have two scenarios: the default time is beyond the
% contract expiry date or is within. In the first case we just compute the
% PV of the cash flows for the whole maturity of the contract. This is
% carried in the main function. In the other case we compute the cash flows
% PV just until the default stopping time.

sum_disc=0;
for i=1:expiry
    if def_time>i                    % if there is no default at time
                                     % "i" we simply add the discount
                                     % factors
        sum_disc=sum_disc+disc_fact(i);
    else
        break                        % in case of default we stop adding.
                                     % The accrual factor is then
                                     % calculated in the other leg of the
                                     % contract
    end
end
PV=sum_disc;
```

default_leg.m

```
function [PV_def,accrued_premium]=default_leg(expiry,def_time,rec,zc_rate)

% This function calculate the present value of the default leg of a basket
% default swap. (The accrued premium is included).
disc_fact_def=0;
last_payment=0;
r=interpolated_zero(def_time,zc_rate);  % in case of a default we calculate
                                        % the corresponding zero rates by
                                        % linear interpolation
disc_fact_def=(1+r)^(-def_time);
PV_def=(1-rec)*disc_fact_def;
```

```
for n=1:expiry
    if def_time<n
        last_payment=n-1;
        break
    end
end
accrued_premium=(def_time-last_payment)*disc_fact_def;
```

basket_spread.m

```
function [a,b,c]=basket_spread(expiry,def_time,rec,zc_rate,discount)

% This function computes the fair spread for each realization of the
% simulated stopping time

% compute the expected premium leg

expected_fees=premium_leg(expiry,def_time,discount);

% compute the expected default leg (default payment + accrued premium)
[exp_rec,accrued]=default_leg(expiry,def_time,rec,zc_rate);

a=expected_fees;
b=exp_rec;
c=accrued;
```

gaussian_time_AV.m

```
function [def_times1 ,def_times2]=gaussian_time_AV(R,hazard,num)

% This function generates default stopping times (generated by a
% multinomial gaussian copula with correlation matrix "r") for the
% obligors included in the basket. Inputs are the correlation matrix
% and the hazard rates for each time and each obligor

% Now we use the simulation algorithm for generating normal variates
% from gaussian copula suggested by ELM(01), pag. 26

n=size(R,1);                % number of obligors in the basket
k=size(hazard,1);           % number of hazard rates available
                            % (corresponding to the number of default
                            % swap premia in the market)
hazard_sum=zeros(k,n);
x1=zeros(num,n);
x2=zeros(num,n);
y=zeros(num,n);
dst1=zeros(num,n);
dst2=zeros(num,n);

A=chol(R); %choleski decomposition
z=randn(num,n); % draw a sample of normal r.v.s. uncorrelated
for i=1:num
y(i,:)=z(i,:)*A;
end
Gauss_U1=normcdf(y);         % 0-1 variates normally distributed
Gauss_U2=normcdf(-y);
% antithetic variate
x1=-log(Gauss_U1);
x2=-log(Gauss_U2);
```

```
for sim=1:num

hazard_sum=zeros(k,n);

o1=zeros(n,1);
o2=zeros(n,1);

for j=1:n                          % for each name in the basket compute the
                                   % default times
    for i=1:k                      % we discretize the integral depicted in
                                   % Schonbucher (2003), pag. 347
        if i==1
            hazard_sum(i,j)=hazard(i,j);
        else
            hazard_sum(i,j)=hazard_sum(i-1,j)+hazard(i,j);
        end
        if hazard_sum(i,j)>=x1(sim,j)
            if i==1
                dst1(sim,j)=x1(sim,j)/hazard(1,j);
                o1(j)=1;
            else
                dst1(sim,j)=((x1(sim,j)-hazard_sum(i-1,j))/hazard(i,j))+
                  (i-1);
                o1(j)=1;
            end
            break
        end
    end
    if hazard_sum(k,j)<=x1(sim,j) & o1(j)~=1
        dst1(sim,j)=((x1(sim,j)-hazard_sum(k-1,j))/hazard(k,j))+(k-1);
    end
end

for j=1:n                          % for each name in the basket compute the
                                   % default times
    for i=1:k                      % we discretize the integral depicted in
                                   % Schonbucher (2003), pag. 347
        if i==1
            hazard_sum(i,j)=hazard(i,j);
        else
            hazard_sum(i,j)=hazard_sum(i-1,j)+hazard(i,j);
        end
        if hazard_sum(i,j)>=x2(sim,j)
            if i==1
                dst2(sim,j)=x2(sim,j)/hazard(1,j);
                o2(j)=1;
            else
                dst2(sim,j)=((x2(sim,j)-hazard_sum(i-1,j))/hazard(i,j))+
                  (i-1);
                o2(j)=1;
            end
```

```
            break
        end
    end
    if hazard_sum(k,j)<=x2(sim,j) & o2(j)~=1
        dst2(sim,j)=((x2(sim,j)-hazard_sum(k-1,j))/hazard(k,j))+(k-1);
    end
end

end
def_times1=dst1;
def_times2=dst2;
```

gaussian_basket_AV.m

```
function [dist,price]=gaussian_basket_AV(T,k,corr,order,hazard,ZC,Recovery)
% This function computes the fair value of a kth to default basket swap
% through simulation by mean of a gaussian copula. With Variance reduction
% scheme (antithetic variates).
% Inputs provided by the users are:
    % T:          basket maturity
    % k:          # simulations
    % R:          correlation matrix
    % order:      1 for 1st to default, 2 for 2nd to default....
    % hazard:     term structure of hazard rate
    % ZC:         zero coupon rates
    % Recovery:   recovery values for each name in the basket
tic

S_fees=0;
S_recovery=0;
S_acc=0;
time1=zeros(size(hazard,2));
index1=zeros(size(hazard,2));
time2=zeros(size(hazard,2));
index2=zeros(size(hazard,2));
t_dist=zeros(size(k,1));


DF=zeros(T,1);
for i=1:T                                    % compute the discount factor
                                             % for each payment date
    for j=1:16                               % loop through all the zero
                                             % rates date knots
        if ZC(j,1)==i;
            DF(i)=(1+ZC(j,2))^(-i);
        end
    end
end
[def_t1, def_t2]=gaussian_time_AV(corr,hazard,k); %load the vector of
                                                  %default times from
                                                  %a normal copula

default=0;
for n=1:k
    if min(def_t1(n,:))<T % count how many defaults
        default=default+1;
    end
```

```
end
defaulter=zeros(default,1);

ttt=1;


for n=1:k                           % start k-th simulation path
    fees1=0;
    recovery1=0;
    acc1=0;
    fees2=0;
    recovery2=0;
    acc2=0;
    [time1,index1]=sort(def_t1(n,:)); %sort the vector
    [time2,index2]=sort(def_t2(n,:));
    tau1=time1(order); % minimum default time
    m1=index1(order); % defaulter
    tau2=time2(order);
    m2=index2(order);
    if tau1<T %default case
        [fees1,recovery1,acc1]=basket_spread(T,tau1,Recovery(m1),ZC,DF);
        defaulter(ttt)=m1;
        ttt=ttt+1;

    else %no default
        for l=1:T
            fees1=fees1+DF(i);
        end
        recovery1=0;
        acc1=0;
    end
    if tau2<T
        [fees2,recovery2,acc2]=basket_spread(T,tau2,Recovery(m2),ZC,DF);
    else
        for l=1:T
            fees2=fees2+DF(i);
        end
        recovery2=0;
        acc2=0;
    end
    fees=0.5*(fees1+fees2);
    recovery=.5*(recovery1+recovery2);
    acc=.5*(acc1+acc2);
    S_fees=S_fees+fees;
    S_recovery=S_recovery+recovery;
    S_acc=S_acc+acc;

end
M_fees=S_fees/k;
M_recovery=S_recovery/k;
M_acc=S_acc/k;
price=(M_recovery/(M_acc+M_fees))*10000;
count1=0;
count2=0;
count3=0;
count4=0;
for g=1:default %count the frequency of default for each obligors
    if defaulter(g)==1
        count1=count1+1;
    elseif defaulter(g)==2
```

```
        count2=count2+1;
    elseif defaulter(g)==3
        count3=count3+1;
    elseif defaulter(g)==4
        count4=count4+1;
    end
end

dist=([default^2 count1 count2 count3 count4]/default)';
toc
```

t_time_AV.m

```
function [def_times1 ,def_times2]=t_time_AV(R,hazard,num,DoF)

% This function generates default stopping times (generated by a
% multinomial t copula with correlation matrix "r" and "DoF" degrees
% of freedom) for the obligors included in the basket. Inputs are the
% correlation matrix and the hazard rates for each time and each obligor

% Now we use the simulation algorithm for generating variates from t
% copula suggested by ELM(01), pag. 26

n=size(R,1);                  % number of obligors in the basket
k=size(hazard,1);             % number of hazard rates available
                              % (corresponding to the number of
                              % default swap premia in the market)
hazard_sum=zeros(k,n);
x1=zeros(num,n);
x2=zeros(num,n);
x=zeros(num,n);
y=zeros(num,n);
dst1=zeros(num,n);
dst2=zeros(num,n);

A=chol(R);
z=randn(num,n);
s=chi2rnd(DoF,num,1);
for i=1:num
y(i,:)=z(i,:)*A;
x(i,:)=(sqrt((DoF/s(i))))*y(i,:);

end

t_U1=tcdf(x,DoF);              % 0-1 variates t-distribuited
t_U2=tcdf(-x,DoF);             % 0-1 variates t-distribuited

x1=-log(t_U1);
x2=-log(t_U2);
for sim=1:num

hazard_sum=zeros(k,n);

o1=zeros(n,1);
o2=zeros(n,1);

for j=1:n                 % for each name in the basket compute the
                          % default times
```

```
    for i=1:k           % we discretize the integral depicted in
                        % Schonbucher (2003), pag. 347
        if i==1
            hazard_sum(i,j)=hazard(i,j);
        else
            hazard_sum(i,j)=hazard_sum(i-1,j)+hazard(i,j);
        end
        if hazard_sum(i,j)>=x1(sim,j)
            if i==1
                dst1(sim,j)=x1(sim,j)/hazard(1,j);
                o1(j)=1;
            else
                dst1(sim,j)=((x1(sim,j)-hazard_sum(i-1,j))/hazard(i,j))+
                  (i-1);
                o1(j)=1;
            end
            break
        end
    end
    if hazard_sum(k,j)<=x1(sim,j) & o1(j)~=1
        dst1(sim,j)=((x1(sim,j)-hazard_sum(k-1,j))/hazard(k,j))+(k-1);
    end
end

for j=1:n               % for each name in the basket compute the
                        % default times
    for i=1:k           % we discretize the integral depicted in
                        % Schonbucher (2003), pag. 347
        if i==1
            hazard_sum(i,j)=hazard(i,j);
        else
            hazard_sum(i,j)=hazard_sum(i-1,j)+hazard(i,j);
        end
        if hazard_sum(i,j)>=x2(sim,j)
            if i==1
                dst2(sim,j)=x2(sim,j)/hazard(1,j);
                o2(j)=1;
            else
                dst2(sim,j)=((x2(sim,j)-hazard_sum(i-1,j))/hazard(i,j))+
                  (i-1);
                o2(j)=1;
            end
            break
        end
    end
    if hazard_sum(k,j)<=x2(sim,j) & o2(j)~=1
        dst2(sim,j)=((x2(sim,j)-hazard_sum(k-1,j))/hazard(k,j))+(k-1);
    end
end

end
def_times1=dst1;
def_times2=dst2;
```

t_basket_AV.m

```
function [dist,price]=t_basket_AV(T,k,corr,order,hazard,ZC,Recovery,DoF)
% This function computes the fair value of a first to default basket swap
% through simulation by mean of a Student's t copula. Inputs provided by
% the users are:
```

```
    % T:          basket maturity
    % k:          # simulations
    % R:          correlation matrix (full)
    % order:      1 for 1st to default, 2 for 2nd to default....
    % hazard:     term structure of hazard rate for each obligor
    % ZC:         zero coupon rates
    % Recovery:   recovery values for each name in the basket
    % DoF:        degrees of freedom of the t copula

tic

S_fees=0;
S_recovery=0;
S_acc=0;
time1=zeros(size(hazard,2));
index1=zeros(size(hazard,2));
time2=zeros(size(hazard,2));
index2=zeros(size(hazard,2));
t_dist=zeros(size(k,1));

DF=zeros(T,1);
for i=1:T                    % compute the discount factor for each
                             % payment date
    for j=1:16               % loop through all the zero rates date knots
        if ZC(j,1)==i;
            DF(i)=(1+ZC(j,2))^(-i);
        end
    end
end
[def_t1, def_t2]=t_time_AV(corr,hazard,k,DoF);

default=0;
for n=1:k
    if min(def_t1(n,:))<T
        default=default+1;
    end
end
defaulter=zeros(default,1);

ttt=1;



for n=1:k                   % start k-th simulation path
    fees1=0;
    recovery1=0;
    acc1=0;
    fees2=0;
    recovery2=0;
    acc2=0;
    [time1,index1]=sort(def_t1(n,:));
    [time2,index2]=sort(def_t2(n,:));
    tau1=time1(order);
    m1=index1(order);
    tau2=time2(order);
    m2=index2(order);
    if tau1<T
        [fees1,recovery1,acc1]=basket_spread(T,tau1,Recovery(m1),ZC,DF);
        defaulter(ttt)=m1;
        ttt=ttt+1;
```

```
    else
        for l=1:T
            fees1=fees1+DF(i);
        end
        recovery1=0;
        acc1=0;
    end
    if tau2<T
        [fees2,recovery2,acc2]=basket_spread(T,tau2,Recovery(m2),ZC,DF);
    else
        for l=1:T
            fees2=fees2+DF(i);
        end
        recovery2=0;
        acc2=0;
    end
    fees=0.5*(fees1+fees2);
    recovery=.5*(recovery1+recovery2);
    acc=.5*(acc1+acc2);
    S_fees=S_fees+fees;
    S_recovery=S_recovery+recovery;
    S_acc=S_acc+acc;


end
M_fees=S_fees/k;
M_recovery=S_recovery/k;
M_acc=S_acc/k;
price=(M_recovery/(M_acc+M_fees))*10000;
count1=0;
count2=0;
count3=0;
count4=0;
for g=1:default
    if defaulter(g)==1
        count1=count1+1;
    elseif defaulter(g)==2
        count2=count2+1;
    elseif defaulter(g)==3
        count3=count3+1;
    elseif defaulter(g)==4
        count4=count4+1;
    end
end

dist=([default^2 count1 count2 count3 count4]/k)';
price=(M_recovery/(M_acc+M_fees))*10000;
toc
```

以下 m—代码从 Excel 表格中载入从欧洲 Libor 自举的零息债券利率和欧洲互换利率.

```
Book = xlsread('C:\ZC\_Rates\ZC.xls');
ZC=zeros(41,2);
ZC(:,1)=Book(:,4);
ZC(:,2)=Book(:,5)/100;
clear Book;
```

5.11 C++中的一揽子信用违约掉期定价

以下C++代码为一揽子信用衍生品的类定义.

BASKET.h

```
#ifndef _BASKET_H__
#define _BASKET_H__
#include "datecl.h"
#include <vector>
#include <string>
#include "time.h"
#include <algorithm>
#include <map>
#include <numeric>
#include "CDO.h"
#ifndef _DEVIATES
#include "MatrixUtil.h"
#endif

#define numSimulations 10000000
#define NOTIONAL 1000000
#define RECOVERY 0.4
#define NUM 4
#define FV 100
#define NUM_SIM 500000
#define step 0.25

static vector<double> hazard;
static vector<int> dayDiff;
static vector<int> matLength;
static vector<double> accrual;
static vector<double> discountRate;
static map<double,double> TR;
static map<double,double> rate;
static int counter = 0;


static void discount(double d) {
  discountRate.push_back(d);
}

static double interpolate(double rate1, double rate2, double t1,
   double t2, double x) {
  double dy = rate2 - rate1;
  double dt = t2 - t1;
  double slope = dy/dt;

  return rate1 + slope*x;
}

class Exposure
{
  public:
      int paySize;
      void setPaySize(double p) { paySize = p; }
      int getPaySize() { return paySize; }
      vector<int> getMatYears() {
          return T;
      }
```

```
vector<double> getHazRate() {
    return hazRate;
}
Exposure(vector<int> matYears, vector<double> spreads,
  string name_, double recovery): T(matYears),
  recoveryRate(recovery), tradeDate("today"),
    notional(NOTIONAL), riskfreeRate(0.02), name(name_) {

    effectiveDate = tradeDate + 1;
    if (effectiveDate == Date::SATURDAY)
        effectiveDate = effectiveDate + 2;
    else if (effectiveDate == Date::SUNDAY)
        effectiveDate = effectiveDate + 1;

    for (int i = 0; i < matYears.size(); i++)
    {
        spreads[i] = (double) spreads[i]/10000;
        Date maturity = effectiveDate.AddYears(matYears[i]);
        maturityDate = computeDate(maturity);
        hazardRate = 0.4;
        spread = spreads[i];
        effectiveDate = tradeDate +  1;
        accrualBasis = computeAccrualBasis(effectiveDate,
           maturityDate);
        accrual.push_back(accrualBasis);
        accrualBasis = accrual[0];
        calcPaymentDates1();
        calibrateHazardRates(i+1);
    }
}
Exposure(Date matDate, double hazRate, double s) :
   recoveryRate(0.4), tradeDate("today"),
    notional(NOTIONAL), riskfreeRate(0.02) {

    effectiveDate = tradeDate + 1;
    if (effectiveDate == Date::SATURDAY)
        effectiveDate = effectiveDate + 2;
    else if (effectiveDate == Date::SUNDAY)
        effectiveDate = effectiveDate + 1;

    if (matDate.day != 20)
        matDate.day = 20;
    if (matDate.day_of_week == Date::SATURDAY)
        matDate.day = 21;
    if (matDate.day_of_week == Date::SUNDAY)
        matDate.day = 22;

    maturityDate = matDate;
    hazardRate = hazRate;
    spread = s;
    accrualBasis = computeAccrualBasis(effectiveDate,maturityDate);
    accrual.push_back(accrualBasis);
    calcPaymentDates();
}
~Exposure() { };
void computePaymentDates(Date start, Date end);
double calcFixedLeg(double h) {

    hazardRate = h;
    int time = computeNumDays(effectiveDate,maturityDate);
```

```
        int time1 = time + 1;
        double survivalProb = (double) exp(-hazardRate*
           ((double)time/365));
        double discountFactor = (double) exp(-riskfreeRate*
           ((double)time1/365));

        return notional*spread*accrualBasis*discountFactor*survivalProb;
    }
    double calcFixedLegDeriv(double h) {

        hazardRate = h;
        int time = computeNumDays(effectiveDate,maturityDate);
        int time1 = time + 1;
        double survivalProb = exp(-hazardRate*((double)time/365));
        double discountFactor = exp(-riskfreeRate*((double)time1/365));

        return (-(double)time/365)*notional*spread*accrualBasis*
          discountFactor*survivalProb;
    }
    double calcNPVFixedTwo(double h) {

        hazardRate = h;
        double survivalProb = exp(-hazard[0]*
          ((double)(matLength[0])/365));
        double survivalProb1 = exp(-hazardRate*
          ((double)(dayDiff[1])/365));
        double discountFactor = exp(-riskfreeRate*
          ((double)(matLength[0] + 1)/365));
        double discountFactor1 = exp(-riskfreeRate*
          ((double)(matLength[1] + 1)/365));

        return notional*spread*accrualBasis*discountFactor*
          survivalProb + notional*spread*((double)dayDiff[1]/360)*
          discountFactor1*survivalProb*survivalProb1;
    }
    double calcNPVFixedTwoDeriv(double h) {

        hazardRate = h;
        double survivalProb = exp(-hazard[0]*
          ((double)(matLength[0])/365));
        double survivalProb1 = exp(-hazardRate*
          ((double)(dayDiff[1])/365));
        double discountFactor = exp(-riskfreeRate*
          ((double)(matLength[1] + 1)/365));

        return -((double)dayDiff[1]/365)*notional*spread*((double)
          dayDiff[1]/360)*discountFactor*survivalProb*survivalProb1;
    }
    double calcNPVFixedThree(double h) {

        hazardRate = h;
        double survivalProb = exp(-hazard[0]*
          ((double)(matLength[0])/365));
        double survivalProb1 = exp(-hazard[1]*((double)(dayDiff[1])/365));
        double survivalProb2 = exp(-hazardRate*
          ((double)(dayDiff[2])/365));
        double discountFactor = exp(-riskfreeRate*
```

```
        ((double)(matLength[0] + 1)/365));
      double discountFactor1 = exp(-riskfreeRate*
        ((double)(matLength[1] + 1)/365));
      double discountFactor2 = exp(-riskfreeRate*
        ((double)(matLength[2] + 1)/365));

      return notional*spread*((double)dayDiff[0]/360)*
        discountFactor*survivalProb
        + notional*spread*((double)dayDiff[1]/360)*
        discountFactor1*survivalProb*survivalProb1
        + notional*spread*((double)dayDiff[2]/360)
        *discountFactor2*survivalProb*survivalProb1*survivalProb2;

  }
  double calcNPVFixedFour(double h) {

      hazardRate = h;
      double survivalProb = exp(-hazard[0]*
        ((double)(matLength[0])/365));
      double survivalProb1 = exp(-hazard[1]*((double)(dayDiff[1])/365));
      double survivalProb2 = exp(-hazard[2]*((double)(dayDiff[2])/365));
      double survivalProb3 = exp(-hazardRate*
        ((double)(dayDiff[3])/365));
      double discountFactor = exp(-riskfreeRate*
        ((double)(matLength[0] + 1)/365));
      double discountFactor1 = exp(-riskfreeRate*
        ((double)(matLength[1] + 1)/365));
      double discountFactor2 = exp(-riskfreeRate*
        ((double)(matLength[2] + 1)/365));
      double discountFactor3 = exp(-riskfreeRate*
        ((double)(matLength[3] + 1)/365));

      return notional*spread*((double)dayDiff[0]/360)*
        discountFactor*survivalProb
        + notional*spread*((double)dayDiff[1]/360)*
        discountFactor1*survivalProb*survivalProb1
        + notional*spread*((double)dayDiff[2]/360)*
        discountFactor2*survivalProb*survivalProb1*survivalProb2
        + notional*spread*((double)dayDiff[3]/360)*discountFactor3*
        survivalProb*survivalProb1*survivalProb2*survivalProb3;

  }
  double calcNPVFixedFive(double h) {

      hazardRate = h;
      double survivalProb = exp(-hazard[0]*
        ((double)(matLength[0])/365));
      double survivalProb1 = exp(-hazard[1]*((double)(dayDiff[1])/365));
      double survivalProb2 = exp(-hazard[2]*((double)(dayDiff[2])/365));
      double survivalProb3 = exp(-hazard[3]*((double)(dayDiff[3])/365));
      double survivalProb4 = exp(-hazardRate*
        ((double)(dayDiff[4])/365));
      double discountFactor = exp(-riskfreeRate*
        ((double)(matLength[0] + 1)/365));
      double discountFactor1 = exp(-riskfreeRate*
        ((double)(matLength[1] + 1)/365));
      double discountFactor2 = exp(-riskfreeRate*
```

```
        ((double)(matLength[2] + 1)/365));
      double discountFactor3 = exp(-riskfreeRate*
        ((double)(matLength[3] + 1)/365));
      double discountFactor4 = exp(-riskfreeRate*
        ((double)(matLength[4] + 1)/365));

      return notional*spread*((double)dayDiff[0]/360)*
        discountFactor*survivalProb
        + notional*spread*((double)dayDiff[1]/360)*
        discountFactor1*survivalProb*survivalProb1
        + notional*spread*((double)dayDiff[2]/360)*
        discountFactor2*survivalProb*survivalProb1*survivalProb2
        + notional*spread*((double)dayDiff[3]/360)*discountFactor3*
        survivalProb*survivalProb1*survivalProb2*survivalProb3
        + notional*spread*((double)dayDiff[4]/360)*discountFactor4*
        survivalProb*survivalProb1*survivalProb2*survivalProb3*
        survivalProb4;
  }
  double calcNPVFixed(double h, int i)
  {
      hazardRate = h;
      double survivalProb = 1.0;
      double discountFactor = 0;
      double sum = 0.0;

      for (int j = 0; j < i; j++)
      {
          discountFactor = exp(-riskfreeRate*((double)(matLength[j] +
            1)/365));
          if (j != i-1)
              survivalProb *= exp(-hazard[j]*
                ((double)(dayDiff[j])/365));
          else
              survivalProb *= exp(-hazardRate*
                ((double)(dayDiff[j])/365));

          sum += notional*spread*((double)dayDiff[i-1]/360)*
            discountFactor*survivalProb;
      }

      return sum;
  }
  double calcNPVFixedThreeDeriv(double h) {

      hazardRate = h;
      double survivalProb = exp(-hazard[0]*
        ((double)(matLength[0])/365));
      double survivalProb1 = exp(-hazard[1]*((double)(dayDiff[1])/365));
      double survivalProb2 = exp(-hazardRate*
        ((double)(dayDiff[2])/365));
      double discountFactor2 = exp(-riskfreeRate*
        ((double)(matLength[2] + 1)/365));

      return-((double)dayDiff[2]/365)*notional*spread*
      ((double)dayDiff[2]/360)*discountFactor2*survivalProb*
      survivalProb1*survivalProb2;
  }
```

```
double calcNPVFixedFourDeriv(double h) {

     hazardRate = h;
     double survivalProb = exp(-hazard[0]*
       ((double)(matLength[0])/365));
     double survivalProb1 = exp(-hazard[1]*((double)(dayDiff[1])/365));
     double survivalProb2 = exp(-hazard[2]*((double)(dayDiff[2])/365));
     double survivalProb3 = exp(-hazardRate*
       ((double)(dayDiff[3])/365));
     double discountFactor3 = exp(-riskfreeRate*
       ((double)(matLength[3] + 1)/365));

     return -((double)dayDiff[3]/365)*notional*spread*((double)
       dayDiff[3]/360)*discountFactor3*survivalProb*survivalProb1*
       survivalProb2*survivalProb3;
}

double calcNPVFixedFiveDeriv(double h) {

     hazardRate = h;
     double survivalProb = exp(-hazard[0]*
       ((double)(matLength[0])/365));
     double survivalProb1 = exp(-hazard[1]*((double)(dayDiff[1])/365));
     double survivalProb2 = exp(-hazard[2]*((double)(dayDiff[2])/365));
     double survivalProb3 = exp(-hazard[3]*((double)(dayDiff[3])/365));
     double survivalProb4 = exp(-hazardRate*
       ((double)(dayDiff[4])/365));
     double discountFactor3 = exp(-riskfreeRate*
       ((double)(matLength[4] + 1)/365));

     return -((double)dayDiff[4]/365)*notional*spread*((double)
       dayDiff[4]/360)*discountFactor3*survivalProb*survivalProb1*
       survivalProb2*survivalProb3*survivalProb4;
}
double calcNPVFixedDeriv(double h, int i) {

     hazardRate = h;
     double survivalProb = 1.0;
     double discountFactor =  exp(-riskfreeRate*
       ((double)(matLength[i-1] + 1)/365));

     for (int j = 0; j < i; j++)
     {
         if (j != i-1)
             survivalProb *= exp(-hazard[j]*
               ((double)(dayDiff[j])/365));
         else
             survivalProb *= exp(-hazardRate*
               ((double)(dayDiff[j])/365));
     }
         return -((double)dayDiff[i-1]/365)*notional*spread*
           discountFactor*survivalProb;

}
double calcNPVFloatingTwo(double h) {

     hazardRate = h;
```

```
        double defaultProb = 1 - exp(-hazard[0]*
          ((double)matLength[0])/365);

        double defaultProb1 = 1 - exp(-hazardRate*
          ((double)dayDiff[1])/365);
        double survivalProb = exp(-hazard[0]*((double)dayDiff[0])/365);
        double discountFactor = exp(-riskfreeRate*
          (double)(matLength[0]+1)/365);
        double discountFactor1 = exp(-riskfreeRate*
          (double)(matLength[1] + 1)/365);
        double accruedInterest1 = 0.5*notional*spread*
          ((double)dayDiff[1]/360);
        double accruedInterest = 0.5*notional*spread*
          ((double)dayDiff[0]/360);
        payments.push_back(accruedInterest);
        payments.push_back(accruedInterest1);

        return (notional*(1 - recoveryRate) - accruedInterest)*
          discountFactor*defaultProb
          + (notional*(1 - recoveryRate) - accruedInterest1)*
          discountFactor1*survivalProb*defaultProb1;

    }
    double calcNPVFloatingThree(double h) {

        hazardRate = h;

        double defaultProb = 1 - exp(-hazard[0]*
          ((double)matLength[0])/365);
        double defaultProb1 = 1 - exp(-hazard[1]*
          ((double)dayDiff[1])/365);
        double defaultProb2 = 1 - exp(-hazardRate*
          ((double)dayDiff[2])/365);

        double survivalProb = exp(-hazard[0]*((double)dayDiff[0])/365);
        double survivalProb1 = exp(-hazard[1]*((double)dayDiff[1])/365);

        double discountFactor = exp(-riskfreeRate*
          (double)(matLength[0]+1)/365);
        double discountFactor1 = exp(-riskfreeRate*
          (double)(matLength[1]+ 1)/365);
        double discountFactor2 = exp(-riskfreeRate*
          (double)(matLength[2]+ 1)/365);

        double accruedInterest = 0.5*notional*spread*
          ((double)dayDiff[0]/360);
        double accruedInterest1 = 0.5*notional*spread*
          ((double)dayDiff[1]/360);
        double accruedInterest2 = 0.5*notional*spread*
          ((double)dayDiff[2]/360);

        payments.push_back(accruedInterest);
        payments.push_back(accruedInterest1);
        payments.push_back(accruedInterest2);

        return (notional*(1 - recoveryRate) - accruedInterest)*
```

```
        discountFactor*defaultProb
        + (notional*(1 - recoveryRate) - accruedInterest1)*
        discountFactor1*survivalProb*defaultProb1
        + (notional*(1 - recoveryRate) - accruedInterest2)*
        discountFactor2*survivalProb*survivalProb1*defaultProb2;
}
double calcNPVFloatingFour(double h)
{

    hazardRate = h;
    double defaultProb = 1 - exp(-hazard[0]*
      ((double)matLength[0])/365);
    double defaultProb1 = 1 - exp(-hazard[1]*
      ((double)dayDiff[1])/365);
    double defaultProb2 = 1 - exp(-hazard[2]*
      ((double)dayDiff[2])/365);
    double defaultProb3 = 1 - exp(-hazardRate*
      ((double)dayDiff[3])/365);

    double survivalProb = exp(-hazard[0]*((double)dayDiff[0])/365);
    double survivalProb1 = exp(-hazard[1]*((double)dayDiff[1])/365);
    double survivalProb2 = exp(-hazard[2]*((double)dayDiff[2])/365);

    double discountFactor = exp(-riskfreeRate*
      (double)(matLength[0]+1)/365);
    double discountFactor1 = exp(-riskfreeRate*
      (double)(matLength[1]+ 1)/365);
    double discountFactor2 = exp(-riskfreeRate*
      (double)(matLength[2]+ 1)/365);
    double discountFactor3 = exp(-riskfreeRate*
      (double)(matLength[3]+ 1)/365);


    double accruedInterest = 0.5*notional*spread*
      ((double)dayDiff[0]/360);
    double accruedInterest1 = 0.5*notional*spread*
      ((double)dayDiff[1]/360);
    double accruedInterest2 = 0.5*notional*spread*
      ((double)dayDiff[2]/360);
    double accruedInterest3 = 0.5*notional*spread*
      ((double)dayDiff[3]/360);

    payments.push_back(accruedInterest);
    payments.push_back(accruedInterest1);
    payments.push_back(accruedInterest2);
    payments.push_back(accruedInterest3);

    return (notional*(1 - recoveryRate) - accruedInterest)*
      discountFactor*defaultProb
      + (notional*(1 - recoveryRate) - accruedInterest1)*
      discountFactor1*survivalProb*defaultProb1
      + (notional*(1 - recoveryRate) - accruedInterest2)*
      discountFactor2*survivalProb*survivalProb1*defaultProb2
      + (notional*(1 - recoveryRate) - accruedInterest3)*
      discountFactor3*survivalProb*survivalProb1*survivalProb2*
      defaultProb3;
```

```
}
double calcNPVFloatingFive(double h)
{

    hazardRate = h;
    double defaultProb = 1 - exp(-hazard[0]*
      ((double)matLength[0])/365);
    double defaultProb1 = 1 - exp(-hazard[1]*
      ((double)dayDiff[1])/365);
    double defaultProb2 = 1 - exp(-hazard[2]*
      ((double)dayDiff[2])/365);
    double defaultProb3 = 1 - exp(-hazard[3]*
      ((double)dayDiff[3])/365);
    double defaultProb4 = 1 - exp(-hazardRate*
      ((double)dayDiff[4])/365);

    double survivalProb = exp(-hazard[0]*((double)dayDiff[0])/365);
    double survivalProb1 = exp(-hazard[1]*((double)dayDiff[1])/365);
    double survivalProb2 = exp(-hazard[2]*((double)dayDiff[2])/365);
    double survivalProb3 = exp(-hazard[3]*((double)dayDiff[3])/365);

    double discountFactor = exp(-riskfreeRate*
      (double)(matLength[0]+1)/365);
    double discountFactor1 = exp(-riskfreeRate*
      (double)(matLength[1]+ 1)/365);
    double discountFactor2 = exp(-riskfreeRate*
      (double)(matLength[2]+ 1)/365);
    double discountFactor3 = exp(-riskfreeRate*
      (double)(matLength[3]+ 1)/365);
    double discountFactor4 = exp(-riskfreeRate*
      (double)(matLength[4]+ 1)/365);

    double accruedInterest = 0.5*notional*spread*
      ((double)dayDiff[0]/360);
    double accruedInterest1 = 0.5*notional*spread*
      ((double)dayDiff[1]/360);
    double accruedInterest2 = 0.5*notional*spread*
      ((double)dayDiff[2]/360);
    double accruedInterest3 = 0.5*notional*spread*
      ((double)dayDiff[3]/360);
    double accruedInterest4 = 0.5*notional*spread*
      ((double)dayDiff[4]/360);

    payments.push_back(accruedInterest);
    payments.push_back(accruedInterest1);
    payments.push_back(accruedInterest2);
    payments.push_back(accruedInterest3);
    payments.push_back(accruedInterest4);

    return (notional*(1 - recoveryRate) - accruedInterest)*
      discountFactor*defaultProb
      + (notional*(1 - recoveryRate) - accruedInterest1)*
      discountFactor1*survivalProb*defaultProb1
      + (notional*(1 - recoveryRate) - accruedInterest2)*
      discountFactor2*survivalProb*survivalProb1*defaultProb2
      + (notional*(1 - recoveryRate) - accruedInterest3)*
      discountFactor3*survivalProb*survivalProb1*
```

```
        survivalProb2*defaultProb3
        + (notional*(1 - recoveryRate) - accruedInterest4)*
        discountFactor4*survivalProb*survivalProb1*survivalProb2*
        survivalProb3*defaultProb4;
  }
  double calcNPVFloating(double h, int i) {

      hazardRate = h;
      double defaultProb = 1.0;
      double discountFactor = 1.0;
      double survivalProb = 1.0;
      double accruedInterest = 0.0;
      double sum = 0.0;

      for (int j = 0; j < i; j++)
      {
          discountFactor *= exp(-riskfreeRate*
            ((double)dayDiff[j]+1)/365);
          survivalProb *= exp(-riskfreeRate*((double)dayDiff[j]+1)/365);
          defaultProb *= 1 - exp(-hazardRate*((double)dayDiff[j])/365);
          accruedInterest = 0.5*notional*spread*
            (((double)dayDiff[j]+1)/360);
          payments.push_back(accruedInterest);
          if (j == 0)
             sum += (notional*(1-recoveryRate) - accruedInterest)*
               discountFactor*defaultProb;
          else
             sum += (notional*(1-recoveryRate) - accruedInterest)*
               discountFactor*survivalProb*defaultProb;
          }

      return sum;

  }
  double calcNPVFloatingTwoDeriv(double h) {

      hazardRate = h;
      double defaultProb = 1 - exp(-hazard[0]*
        ((double)matLength[0])/365);
      double defaultProb1 = 1 - exp(-hazardRate*
        ((double)dayDiff[1])/365);

      double survivalProb = exp(-hazard[0]*((double)dayDiff[0])/365);
      double discountFactor1 = exp(-riskfreeRate*
        (double)(matLength[1] + 1)/365);
      double accruedInterest = 0.5*notional*spread*
        ((double)dayDiff[1]/360);

      double z = ((double)dayDiff[1]/365)*
        (notional*(1 - recoveryRate) - accruedInterest)*
          discountFactor1*survivalProb*defaultProb1;

      return z;
  }
  double calcNPVFloatingThreeDeriv(double h) {

      hazardRate = h;
```

```
    double defaultProb2 = 1 - exp(-hazardRate*
      ((double)dayDiff[2])/365);

    double survivalProb =  exp(-hazard[0]*((double)dayDiff[0])/365);
    double survivalProb1 = exp(-hazard[1]*((double)dayDiff[1])/365);
    double discountFactor2 = exp(-riskfreeRate*
      (double)(matLength[2] + 1)/365);
    double accruedInterest2 = 0.5*notional*spread*
      ((double)dayDiff[2]/360);

    double z = ((double)dayDiff[2]/365)*
      (notional*(1 - recoveryRate) - accruedInterest2)*
        discountFactor2*survivalProb*survivalProb1*defaultProb2;

    return z;
}
double calcNPVFloatingFourDeriv(double h) {

    hazardRate = h;
    double defaultProb3 = 1 - exp(-hazardRate*
      ((double)dayDiff[3])/365);

    double survivalProb =  exp(-hazard[0]*((double)dayDiff[0])/365);
    double survivalProb1 = exp(-hazard[1]*((double)dayDiff[1])/365);
    double survivalProb2 = exp(-hazard[2]*((double)dayDiff[2])/365);

    double discountFactor3 = exp(-riskfreeRate*
      (double)(matLength[3] + 1)/365);
    double accruedInterest3 = 0.5*notional*spread*
      ((double)dayDiff[3]/360);

    return ((double)dayDiff[3]/365)*(notional*(1 - recoveryRate) -
      accruedInterest3)*discountFactor3*survivalProb*survivalProb1*
      survivalProb2*defaultProb3;
}
double calcNPVFloatingFiveDeriv(double h) {

    hazardRate = h;
    double defaultProb4 = 1 - exp(-hazardRate*
      ((double)dayDiff[4])/365);

    double survivalProb =  exp(-hazard[0]*((double)dayDiff[0])/365);
    double survivalProb1 = exp(-hazard[1]*((double)dayDiff[1])/365);
    double survivalProb2 = exp(-hazard[2]*((double)dayDiff[2])/365);
    double survivalProb3 = exp(-hazard[3]*((double)dayDiff[3])/365);

    double discountFactor4 = exp(-riskfreeRate*
      (double)(matLength[4] + 1)/365);
    double accruedInterest4 = 0.5*notional*spread*
      ((double)dayDiff[4]/360);

    return ((double)dayDiff[4]/365)*(notional*(1 - recoveryRate)  -
      accruedInterest4)*discountFactor4*survivalProb*survivalProb1*
      survivalProb2*survivalProb3*defaultProb4;

}
```

```
double calcFloatingLegDeriv(double h) {

    hazardRate = h;
    double accruedInterest = calcRebate(effectiveDate);
    int time = computeNumDays(effectiveDate,maturityDate);
    int time1 = time + 1;
    double defaultProb = 1 - exp(-hazardRate*((double)time/365));
    double discountFactor = exp(-riskfreeRate*((double)time1/365));

    return ((double)time/365)*(notional*(1 - recoveryRate) -
      accruedInterest)*discountFactor*defaultProb;

}
double calcNPVFloatingDeriv(double h, int i)
{

    hazardRate = h;
    double defaultProb = 1 - exp(-hazardRate*
      ((double)dayDiff[i-1])/365);
    double discountFactor = exp(-riskfreeRate*
      ((double)matLength[i-1] + 1)/365);
    double accruedInterest = 0.5*notional*spread*
      ((double)dayDiff[i-1]/360);
    double survivalProb = 1;

    if (i != 1)
    {
        for (int j = 0; j < i; j++)
        {
            survivalProb *= exp(-hazard[j]*((double)dayDiff[j])/365);
        }
    }

    return ((double)dayDiff[i-1]/365)*(notional*(1 - recoveryRate) -
      accruedInterest)*discountFactor*survivalProb*defaultProb;
}
double calcFloatingLeg(double h) {

    hazardRate = h;
    double accruedInterest = calcRebate(effectiveDate);
    payments.push_back(accruedInterest);
    int time = computeNumDays(effectiveDate,maturityDate);
    int time1 = time + 1;
    double defaultProb = (double) 1 - exp(-hazardRate*
      ((double)time/365));
    double discountFactor = exp(-riskfreeRate*((double) time1/365));

    return (notional*(1 - recoveryRate) - accruedInterest)*
      discountFactor*defaultProb;
}
int computeNumDays(Date start, Date end1) {

    return end1 - start + 1;
}
void setNumDays(int numDay) {

    numDays = numDay;
```

```
    }
    double calcRebate(Date defaultDate) {

        int time1 = maturityDate - effectiveDate;
        return 0.5*notional*spread*((double)time1/360);

    }
    double calcRebate1() {

        int time1 = dayDiff[1];
        return 0.5*notional*spread*((double)time1/360);
    }
    void calcPaymentDates()
    {
        Date d, paymentDay;
        int cnt = 1;
        payments.empty();
        payments.clear();

        std::cout << "payment size = " << payments.size() << "cnt = " <<
          cnt <<  endl;
        Date firstDate = computeDate(effectiveDate);
        std::cout << "first Payment Date " << firstDate << endl;
        paymentDates.push_back(firstDate);
        int diff = firstDate - effectiveDate + 1;

        periodLength.push_back(diff);
        payments.push_back(notional*spread*((double)diff/365));
        survival.push_back(exp(-hazardRate*((double)diff/365)));

        do
        {
            if (cnt != 1)
                d = firstDate.AddMonths(3);
            else
                d = firstDate;
            paymentDay = computeDate(d);
            paymentDates.push_back(paymentDay);
            int length = paymentDates[cnt] - paymentDates[cnt-1];
            int duration = paymentDates[cnt] - tradeDate + 1;
            survival.push_back(exp(-hazardRate*((double)duration/365)));
            double val = (notional*spread*((double)length/365))*
              survival[cnt-1]*discountRate[cnt-1];
            std::cout << "val = " << val << endl;
            payments.push_back(val);
            periodLength.push_back(length);
            cnt++;
        }
        while (paymentDay != maturityDate);

            totalPay = accumulate(payments.begin(),payments.end(),
              (float)0.0);
            std::cout << "payment size = " << payments.size() <<
              "cnt = " << cnt <<  endl;
            setPaySize(payments.size());
            payments.empty();
            payments.clear();
```

```
}
void calcPaymentDates1()
{
    Date d, paymentDay;
    int cnt = 1;
    payments.empty();
    payments.clear();

    Date firstDate = computeDate(effectiveDate);
    paymentDates.push_back(firstDate);
    int diff = firstDate - effectiveDate + 1;
    periodLength.push_back(diff);
    payments.push_back(notional*spread*((double)diff/365));
    survival.push_back(exp(-hazardRate*((double)diff/365)));

    do
    {
        if (cnt != 1)
            d = firstDate.AddMonths(3);
        else
            d = firstDate;
        paymentDay = computeDate(d);
        paymentDates.push_back(paymentDay);
        int length = paymentDates[cnt] - paymentDates[cnt-1];
        int duration = paymentDates[cnt] - tradeDate + 1;
        survival.push_back(exp(-hazardRate*((double)duration/365)));
        double val = (notional*spread*((double)length/365))*
          survival[cnt-1]*discountRate[cnt-1];
        payments.push_back(val);
        periodLength.push_back(length);
        cnt++;
    }
    while (paymentDay <= maturityDate);

    totalPay = accumulate(payments.begin(),payments.end(),(float)0.0);
    setPaySize(payments.size());
    payments.empty();
    payments.clear();
}
Date computeDate(Date maturity)
{
    if ((maturity.month == Date::JANUARY) || (maturity.month ==
      Date::FEBRUARY))
    {
        maturity.month = Date::MARCH;
        maturity.day = 20;
        if (maturity.day_of_week == Date::SATURDAY)
            maturity.day = 21;
        if (maturity.day_of_week == Date::SUNDAY)
            maturity.day = 22;
        }
        else if (maturity.month == Date::MARCH)
        {
            if (maturity.day < 20)
            {
                maturity.day = 20;
```

```
            if (maturity.day_of_week == Date::SATURDAY)
                maturity.day = 21;
            if (maturity.day_of_week == Date::SUNDAY)
                maturity.day = 22;
        }
        else if (maturity.day != 20)
        {
            maturity.month = Date::JUNE;
            maturity.day = 20;
            if (maturity.day_of_week == Date::SATURDAY)
                maturity.day = 21;
            if (maturity.day_of_week == Date::SUNDAY)
                maturity.day = 22;
        }
    }
    else if ((maturity.month == Date::APRIL) ||
      (maturity.month == Date::MAY))
    {
        maturity.month = Date::JUNE;
        maturity.day = 20;
        if (maturity.day_of_week == Date::SATURDAY)
            maturity.day = 21;
        if (maturity.day_of_week == Date::SUNDAY)
            maturity.day = 22;
    }
    else if (maturity.month == Date::JUNE)
    {
        if (maturity.day < 20)
        {
            maturity.day = 20;
            if (maturity.day_of_week == Date::SATURDAY)
                maturity.day = 21;
            if (maturity.day_of_week == Date::SUNDAY)
                maturity.day = 22;
            }
            else if (maturity.day != 20)
            {
                maturity.month = Date::SEPTEMBER;
                maturity.day = 20;
                if (maturity.day_of_week == Date::SATURDAY)
                    maturity.day = 21;
                if (maturity.day_of_week == Date::SUNDAY)
                    maturity.day = 22;
            }
        }
        else if ((maturity.month == Date::JULY) ||
          (maturity.month == Date::AUGUST))
        {
            maturity.month = Date::SEPTEMBER;
            maturity.day = 20;
            if (maturity.day_of_week == Date::SATURDAY)
                maturity.day = 21;
            if (maturity.day_of_week == Date::SUNDAY)
                maturity.day = 22;
        }
        else if (maturity.month == Date::SEPTEMBER)
        {
```

```
            if (maturity.day < 20)
            {
                maturity.day = 20;
                if (maturity.day_of_week == Date::SATURDAY)
                    maturity.day = 21;
                if (maturity.day_of_week == Date::SUNDAY)
                    maturity.day = 22;
                }
                else if (maturity.day != 20)
                {
                    maturity.month = Date::DECEMBER;
                    maturity.day = 20;
                    if (maturity.day_of_week == Date::SATURDAY)
                        maturity.day = 21;
                    if (maturity.day_of_week == Date::SUNDAY)
                        maturity.day = 22;
                }
        }
        else if ((maturity == Date::OCTOBER) || (maturity ==
          Date::NOVEMBER))
        {
            maturity.month = Date::DECEMBER;
            maturity.day = 20;
            if (maturity.day_of_week == Date::SATURDAY)
                maturity.day = 21;
            if (maturity.day_of_week == Date::SUNDAY)
                maturity.day = 22;
        }
        else if (maturity.month == Date::DECEMBER)
        {
            if (maturity.day < 20)
            {
                maturity.day = 20;
                if (maturity.day_of_week == Date::SATURDAY)
                    maturity.day = 21;
                if (maturity.day_of_week == Date::SUNDAY)
                    maturity.day = 22;
            }
            else if (maturity.day != 20)
            {
                maturity.month = Date::MARCH;
                maturity.day = 20;
                if (maturity.day_of_week == Date::SATURDAY)
                    maturity.day = 21;
                if (maturity.day_of_week == Date::SUNDAY)
                    maturity.day = 22;
                maturity.year = maturity.year + 1;
            }
        }
        return maturity;
}
double computeAccrualBasis(Date start, Date end) {

    // compute accrual basis
    numDays = computeNumDays(effectiveDate,maturityDate);
    setNumDays(numDays);
    matLength.push_back(numDays);
```

```
        counter++;
        if (dayDiff.size() == 0)
            dayDiff.push_back(numDays);
        else
        {
            int diff1 = matLength[counter-1] - matLength[counter-2] + 1;
            dayDiff.push_back((int)diff1);
        }
        accrualBasis = (double) numDays/360;
        return accrualBasis;
    }
    void setMaturityDate(Date d1)
    {
        if ((d1 != Date::JUNE) && (d1 != Date::SEPTEMBER)
            && (d1 != Date::DECEMBER) && (d1 != Date::MARCH))
        {
            //cout << "Maturity month must be March, June, Sept., or
              Dec." << endl;
            maturityDate = "6/20/2009";  // default maturity
        }
        else
            maturityDate = d1;

    }
    void calibrateHazardRates(int num)
    {

        int j = 0;
        double h = hazardRate;
        double f, fd;
        double error = 0.0;
        double dfixedleg = 0.0;
        double dfloatingleg = 0.0;

        switch (num)
        {
        case 1:
            do
            {
                f = calcFloatingLeg(h) - calcFixedLeg(h);
                dfixedleg = calcFixedLegDeriv(h);
                dfloatingleg = calcFloatingLegDeriv(h);
                fd = dfloatingleg - dfixedleg;
                h = h - 0.01*(f/fd);
                error = -0.01*f/fd;
            }
            while (fabs(error) > 0.00001);
            break;
        case 2:
            do
            {
                f = calcNPVFloatingTwo(h) - calcNPVFixedTwo(h);
                dfixedleg = calcNPVFixedTwoDeriv(h);
                dfloatingleg = calcNPVFloatingTwoDeriv(h);
                fd = dfloatingleg - dfixedleg;
                h = h - 0.01*(f/fd);
                error = -0.01*f/fd;
```

```
                }
                while (fabs(error) > 0.00001);
                break;
            case 3:
                do
                {
                    f = calcNPVFloatingThree(h) - calcNPVFixedThree(h);

                    dfixedleg = calcNPVFixedThreeDeriv(h);
                    dfloatingleg = calcNPVFloatingThreeDeriv(h);
                    fd = dfloatingleg - dfixedleg;
                    h = h - 0.01*(f/fd);
                    error = -0.01*f/fd;
                }
                while (fabs(error) > 0.00001);
                break;
            case 4:
                do
                {
                    f = calcNPVFloatingFour(h) - calcNPVFixedFour(h);

                    dfixedleg = calcNPVFixedFourDeriv(h);
                    dfloatingleg = calcNPVFloatingFourDeriv(h);
                    fd = dfloatingleg - dfixedleg;
                    h = h - 0.01*(f/fd);
                    error = -0.01*f/fd;
                }
                while (fabs(error) > 0.00001);
                break;
            case 5:
                do
                {
                    f = calcNPVFloatingFive(h) - calcNPVFixedFive(h);
                    dfixedleg = calcNPVFixedFiveDeriv(h);
                    dfloatingleg = calcNPVFloatingFiveDeriv(h);
                    fd = dfloatingleg - dfixedleg;
                    h = h - 0.01*(f/fd);
                    error = -0.01*f/fd;
                }
                while (fabs(error) > 0.00001);
                 break;
            }
            hazardRate = h;
            hazard.push_back(hazardRate);
            hazRate.push_back(hazardRate);
            std::cout << getName() << " " << T[num-1] <<
              "year  hazardRate = " << hazardRate << endl;
    }
    void setName(string name_) { name = name_; }
        string getName() { return name; }
private:
   friend class Basket;
   double hazardRate;
   vector<double> hazRate;
   map<Date,double> matHaz;
   double accrualBasis;
   double totalPay;
```

```
    Date defaultTime;
    vector<int> periodLength;
    vector<Date> paymentDates;
    vector<double> payments;
    vector<double> survival;
    double survivalRate;
    double recoveryRate;
    double riskfreeRate;
    string name;
    Date maturityDate;
    double notional;
    int numDays;
    Date tradeDate;  // start date
    Date effectiveDate;
    double spread;
    double rebate;
    vector<int> T;
};

struct BondPrice
{
    double price;
    double coupon;
    int numPeriods;
};

class Basket
{
  public:
      Basket() { kthToDefault = 1; }
      ~Basket() {};
      void bootstrap(vector<double> price, vector<double> coupon,
        vector<double> period);
      void bootstrap(map<double,double> TR);
      double genCorrelatedDeviates();
      double priceBasket();
      double priceCDOBasket(int numReference, int expiry,
        vector<Tranche> tranche);
      void computePaymentDates(Date start, Date end);
      void setEffectiveDate(Date d1);
      void setTradeDate(Date d1);
      void setMaturityDate(Date d1);
      void setkthtoDefault(int kthDefault);
      void addExposure(Exposure& ex) {
          exposure.push_back(ex);
      }
      double calcSumDiscPayments(Date d, int j);
      double calcDefaultTime(double d, int j);
      int getNumExposures();
      void printHazard() {

          int num = exposure.size();
          vector<double>::iterator iter;
          vector<double> v;
          vector<int> year = exposure[0].getMatYears();

          std::cout << "              ";
```

```
            for (int i = 0; i < num; i++)
                std::cout << exposure[i].getName() << "       "
            std::cout << endl << endl;

            for (i = 0; i < num; i++) {
                std::cout << "Year " << year[i] << "     ";
                v = exposure[i].getHazRate();
                for (iter = v.begin(); iter != v.end(); iter++) {
                    std::cout << "    " << *iter;
                }
                std::cout << endl;
            }
        }
private:
   vector<Exposure> exposure;
   int frequency;          // frequency of payments
   Date tradeDate;         // start date
   Date effectiveDate;
   Date maturityDate;      // maturity of basket
   int numExposures;       // number of exposures in the basket
   vector<Date> paymentDates;
   vector<double> spotRates;
   vector<Date> numDays;
   vector<double> cdsSpreads;
   vector<double> termStructure;
   vector<double> coupons;
   vector<double> bondPrice;
   vector<double> period;
   int kthToDefault;
   vector<double> discountRate;
   vector<double> discountSum;
   double faceValue;
};

#endif
```

函数定义如下：

BASKET.cpp

```
#include "Basket.h"
#include <map>
#define NUM 5
#define step 0.25

void Basket::bootstrap(vector<double> price, vector<double> coupon,
  vector<double> period)
{
        copy(price.begin(),bondPrice.begin(),bondPrice.end());
        copy(coupon.begin(),coupons.begin(),coupons.end());
        copy(period.begin(),period.begin(),period.end());

        spotRates[0] = (((coupons[0] + faceValue)/bondPrice[0])-1)*2;
        discountSum[0] = pow(1/(1 + spotRates[0]),period[0]);

        for (int i = 1; i <= bondPrice.size(); i++)
        {
                spotRates[i] = 2*(pow((coupons[i] +
                  faceValue)/(bondPrice[i] - coupons[i]*discountSum[i-1]),
```

```
                (1/period[i]))-1);
            discountSum[i] = discountSum[i-1] +
              pow(1/(1 + spotRates[i]),period[i]);
        }
}

void Basket::setkthtoDefault(int kthDef) {
        kthToDefault = kthDef;
}
void Basket::setTradeDate(Date d1) {
        tradeDate = d1;
}

void Basket::setEffectiveDate(Date d1) {
        effectiveDate = d1;
}

void Basket::setMaturityDate(Date d1) {
        maturityDate = d1;
}

double Basket::priceBasket()
{
        int i, j, k;
        MRNG mrng;  // Mersenne generator
        MatrixUtil mu;  // Matrix Utility
        double* z = new double[NUM];
        double aveInflow = 0.0;
        double aveOutflow = 0.0;
        vector<double> defaultTime;
        vector<double> normaldev;
        TNT::Array1D<double> dev(NUM);
        double minTime = 0.0;
        double basketPrice = 0.0;
        double sum, outflow = 0.0;
        double inflow = 0.0;      // cash inflows
        double y = 0.0;           // default time
        int m = exposure.size();  // number of exposure in the basket
        int def = 0;
        double total_protection_leg = 0.0;
        double total_fixed_leg = 0.0;
        Array2D<double> R(NUM,NUM);
        map<double,int> expMap;
        std::cout.precision();

        for (i = 0; i < NUM; i++)
        {
                for (j = 0; j < NUM; j++)
                {
                        if (i == j)
                                R[i][j] = 1;
                        else
                                // flat correlation
                                R[i][j] = R[j][i] = 0.62;
                        }
                }
        }

        // sample correlation structure
        /*
```

```
		R[0][1] = R[1][0] = 0.44180;
		R[0][2] = R[2][0] = 0.90208;
		R[0][3] = R[3][0] = 0.83975;
		R[1][2] = R[2][1] = 0.67615;
		R[1][3] = R[3][1] = 0.68552;
		R[2][3] = R[3][2] = 0.84178;
		*/

		for (i = 1; i <= NUM_SIM; i++)
		{
			sum = 0.0;
			normaldev.empty();
			defaultTime.empty();
			normaldev.clear();
			defaultTime.clear();

			// for each name in the basket
			for (j = 0; j < m; j++)
			{
				dev = mu.genCholesky(R);
				normaldev.push_back(mu.normalCalc(dev[j]));
				y = calcDefaultTime(-log(normaldev[j]),j);
				defaultTime.push_back(y);
				expMap[defaultTime[j]] = j;
			}
			vector<double>::iterator iter;
			vector<double>::iterator defIter;

			sort(defaultTime.begin(),defaultTime.end());
			defIter = defaultTime.begin();
			minTime = *defIter;

			switch (kthToDefault)
			{
				case 1:
				    {
				  k = expMap[minTime];
				  break;
				    }
				case 2:
				    {
				  defIter++;
				  minTime = *defIter;
				  k = expMap[minTime];
				  break;
				    }
				case 3:
				    {
				  defIter++;
				  defIter++;
				  minTime = *defIter;
				  k = expMap[minTime];
				  break;
				    }
				default:
				  k = expMap[minTime];
				  break;
			}

			Date effectiveDate = "today";
```

```
                int months = (int) (minTime - (minTime/100))*12;
                int years = (int) minTime / 100;
                Date defTime = effectiveDate.AddYears(years);
                defTime = effectiveDate.AddMonths(months);
                int diff = defTime - exposure[k].effectiveDate;

                if (defTime > exposure[k].maturityDate)
                {
                // inflow = inflow + 0;
                // receive nothing
                total_fixed_leg = exposure[k].totalPay;
                outflow = outflow + total_fixed_leg;
                // pay total discounted spread on notional
                // = sum of notional*(num days/360)*(spread)
                }
        else
        {

                    total_protection_leg = ((exposure[k].notional)*(1 -
                      exposure[k].recoveryRate))*exp(-
                      (exposure[k].riskfreeRate)*((double)diff/365));
                    inflow = inflow + total_protection_leg;
                    // sum of discounted protection payments up to
                    // time of default
                    // = (value of 1 basis pt)*(min default
                    // time of all names in basket)
                    outflow = outflow + calcSumDiscPayments(defTime,k);

                }
        }
        std::cout << "inflow = " << inflow << endl;
        std::cout << "outflow = " << outflow << endl;
        aveInflow = (double) inflow/M;
        aveOutflow = (double) outflow/M;

        basketPrice = (aveInflow/aveOutflow)*(exposure[k].spread)*10000;
        std::cout << "basketPrice = " << basketPrice << endl;

        return 0;
}


double Basket::calcSumDiscPayments(Date defDate, int j)
{
        double sum = 0.0;
        double discPay = 0.0;

        for (int i = 0; i < exposure[j].getPaySize(); i++)
        {

                if (exposure[j].paymentDates[i] < defDate)
                {
                        int length = defDate - exposure[j].paymentDates[i];
                        discPay = exposure[j].payments[i]*
                          exp(-(exposure[j].riskfreeRate)*
                          ((double)length/365));
                        sum = sum + discPay;
                }
        }
        return sum;
}
```

```
int Basket::getNumExposures() {
        return numExposures;
}

double Basket::calcDefaultTime(double dev, int j) {

        double tau = 0.0;
        double hazard_sum[10];
        double o[10] = {0.0};

        int k = exposure[0].hazRate.size()-1;
        int i = 0;
        double y = 0.0;

        for (i = 0; i <= k; i++)
        {
                if (i == 0)
                        hazard_sum[0] = exposure[j].hazRate[0];
                else
                        hazard_sum[i] = hazard_sum[i-1] +
                          exposure[j].hazRate[i];

                if (hazard_sum[i] >= dev)
                {
                        if (i == 0)
                        {
                                tau = dev/exposure[j].hazRate[0];
                                o[j] = 1.0;
                        }
                        else
                        {
                                tau = ((dev - hazard_sum[i-1])/
                                  (exposure[j].hazRate[i]))+i;
                                o[j] = 1.0;
                        }
                        break;
                }
        }
        if ((hazard_sum[k] <= dev) && (o[j] != 1))
                tau = ((dev - hazard_sum[k-1])/
                  (exposure[j].hazRate[k])) + k;

        return tau;
}
```

为了计算相关性偏离，我们使用一个矩阵应用类 MatrixUtil，这个类利用了数值工具箱模板（TNT）的矩阵函数来计算 Cholesky 分解. 这个类还用到了 Random 类中的 Mersenne Twister 随机数生成器，生成随机偏移.

假设要为一个 5 年期、4 个参考实体的篮子进行定价，其利差如下：

	Maturity(yrs)				
	1	2	3	4	5
Apple	100	150	160	170	185
IBM	75	120	200	220	250
GM	400	450	500	530	580
Northwest	420	450	500	520	550

相关性结构为：

1	0.44180	0.90208	0.83975
0.44180	1	0.67615	0.25903
0.90208	0.67615	1	0.84178
0.83975	0.25903	0.841878	1

主函数如下：

BasketPricing_Main.cpp

```
#include <strstream>
#include <iostream>
#include <fstream>
#include "Basket.h"
#define SIZE_X 1000
using namespace std;

void main()
{

  Basket basket;
  basket.setEffectiveDate("today");
  char buffer[SIZE_X];
  char dataBuffer[SIZE_X];
  char* str = NULL;
  double disc = 0.0;
  // make sure the path to the folder is correct
  const char* file = "c:\\BasketCDSPricing\\Discount.txt";

  ifstream fin;              // input file stream
  fin.clear();
  fin.open(file);

  if (fin.good())
  {
        while (!fin.eof())
        {
          fin.getline(buffer,sizeof(buffer)/sizeof(buffer[0]));
          istrstream str(buffer);
          // Get data
          str >> dataBuffer;
          disc = atof(dataBuffer);
          discount(disc);
        }
  }
  else
              std::cout << "File not good!" << "\n";

  fin.close();

  vector<int> expDates1;
  expDates1.push_back(1);
  expDates1.push_back(2);
  expDates1.push_back(3);
  expDates1.push_back(4);
  expDates1.push_back(5);
```

```
    vector<double> s1, s2, s3, s4, s5;
    s1.push_back(100);
    s1.push_back(150);
    s1.push_back(160);
    s1.push_back(170);
    s1.push_back(185);

    s2.push_back(75);
    s2.push_back(120);
    s2.push_back(200);
    s2.push_back(220);
    s2.push_back(250);

    s3.push_back(400);
    s3.push_back(450);
    s3.push_back(500);
    s3.push_back(530);
    s3.push_back(580);

    s4.push_back(420);
    s4.push_back(450);
    s4.push_back(500);
    s4.push_back(520);
    s4.push_back(550);

    std::cout << "Calibrating Hazard Rates..." << endl << endl;

    Exposure exp1(expDates1,s1,"Apple",0.4);
    Exposure exp2(expDates1,s2,"IBM",0.4);
    Exposure exp3(expDates1,s3,"GM",0.4);
    Exposure exp4(expDates1,s4,"Northwest",0.4);

    basket.addExposure(exp1);
    basket.addExposure(exp2);
    basket.addExposure(exp3);
    basket.addExposure(exp4);

    std::cout << endl;
    std::cout << "Pricing Basket..." << endl;
    //basket.setkthtoDefault(2);
    basket.priceBasket();
}
```

输出如下：

```
Calibrating Hazard Rates...

Apple 1year hazardRate = 0.0169168
Apple 2year hazardRate = 0.0345374
Apple 3year hazardRate = 0.0307447
Apple 4year hazardRate = 0.0344349
Apple 5year hazardRate = 0.0431177

IBM 1year hazardRate = 0.0126858
IBM 2year hazardRate = 0.0239344
IBM 3year hazardRate = 0.0525948
IBM 4year hazardRate = 0.0720684
IBM 5year hazardRate = 0.107577
```

```
GM 1year hazardRate = 0.0677314
GM 2year hazardRate = 0.14406
GM 3year hazardRate = 0.228336
GM 4year hazardRate = 0.334723
GM 5year hazardRate = 0.502378

Northwest 1year hazardRate = 0.0711293
Northwest 2year hazardRate = 0.14406
Northwest 3year hazardRate = 0.228336
Northwest 4year hazardRate = 0.325314
Northwest 5year hazardRate = 0.460316

basketPrice = 784.92
```

需要注意的是，由于相关性结构的关系，整个篮子价格要小于分别对每一参考实体都购买5年期信用违约掉期保护的价格.实际上，大部分交易台都用平坦相关性结构来为篮子进行定价.例如，假设所有贷款人的违约相关性都是0.466，篮子的价格就是785.23个基点.

5.12 信用联系票据（CLN）

信用联系票据使得投资人对于其购买或提供融资的资产，获得与该项资产本身的信用风险，再加上一个通过银行和票据发行方之间的信用违约掉期而传递的信用风险相关的回报.信用联系票据使得用信用违约掉期所转移的风险嵌入一个证券之中并向投资人发行.[31]除非标的信用发生了某个信用事件，否则投资人都能得到票息并领取对票面价值的赎回金.如果有信用事件的发生，赎回金额就等于票面价值减掉一个或有偿付款.[32]银行和票据发行人之间的交易是一个信用违约掉期，如图5.10所示.

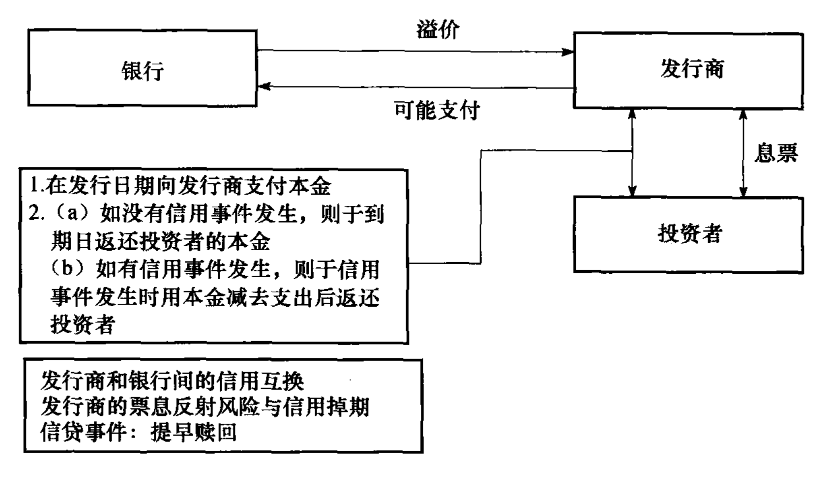


图5.10　CLN交易

投资人承担两个主要风险：其一，投资人暴露于信用违约的风险（通过票据发行人和银行间的信用违约掉期）；其二，就是中期票据（MTN）发行者的信用价值的风险.如果参考信用发生违约，该事件就会造成票据的提前赎回和减价赎回.如果发行人发生违约，那么投资人就暴露于证券回收以及信用违约掉期市场价格的风险.对于这两类风险的暴露反映在了票据的收

益上. 该票据的收益等于从票据发行人处得到的回报和信用违约掉期溢价之和. 一个只存在于纸面的无形公司——也被称为特设机构（SPV），可以为投资人减少所承担的风险. 这个公司的信用评级通常是AAA，通过在一个结构性票据内打包一个信用衍生品，为参考信用风险提供保护. 因此，信用联系票据可以视为由SPV进行打包并对其售出一个保险期权的信用资产，这个信用资产可以是贷款或债券，这个保险期权则往往是一个信用违约期权或总收益掉期. SPV由某个母公司为了获得对标的资产持有而特别设立. 该参考资产与其他SPV内部的资产没有联系，SPV随后出售一个信用违约期权来对其提供违约保护. 发行人由于暴露于参考信用的信用风险会收到一笔溢价. 这笔溢价构成了向投资人支付的票息的一部分. 当参考资产发生了一个信用事件，票据被提前赎回，发行人向银行支付一笔或有偿付款，其余额支付给投资人. 由于在SPV层面没有税收问题，卖出期权获得的溢价和信用资产现金流的回报直接流向购买了信用联系票据的投资人. 由于缺乏基建和流动性，通常投资人没有机会接触到信用风险，而信用联系票据为投资人提供了承担信用风险的渠道. 比如，一个在美国没有任何海外银行联系的投资人可以通过打包了某笔贷款的信用联系票据来投资日本银行的贷款资产，如图5.11所示.

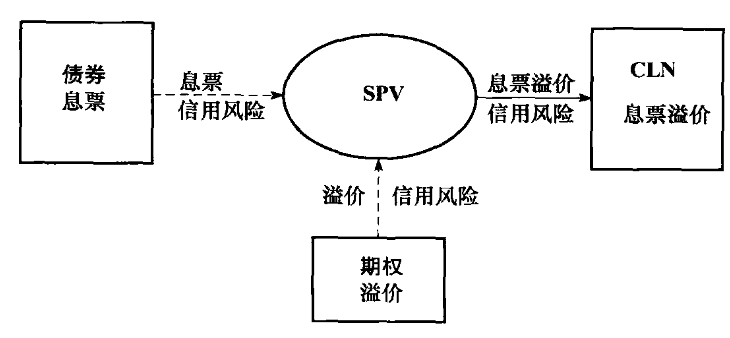


图5.11 CLN结构

由于信用联系票据的收益取决于两个不同的资产，于是在两种情况下该票据会发生违约：评级较高的资产的违约——尽管概率很小，但还是有可能发生；或者是卖出期权的参考资产违约. 如果参考资产发生违约，SPV通常会卖出先前购买的高评级的资产，将其部分收入用来偿还保护购买人的违约损失. 其余的收入流向信用联系票据的投资人，他们只能获得原先投资的一部分. 到了这一步，这个信用联系票据终止.

信用联系票据的价格通常以对某个基准的利差进行报价，如LIBOR或UST. 这个价格等于SPV获得的所有现金流. 例如，如果SPV持有债券并获得信托收益为UST+20bps，它还持有一项信用违约期权的空头头寸，溢价为295bps，那么这个将信用衍生品和债券相联系的票据价格为UST+315bps，如图5.12所示.

通过将以一个高评级资产为担保的证券和一个信用违约期权结合在一起，机构可以创造出一个相比于同类资产市场价格更便宜（换言之，有着更高的收益）的合成债券. 违约溢价加上资产抵押证券的回报使得信用联系票据的收益要高于该资产债券本身. 例如，10年期的名义价值为100万美元的汽车制造商债券以LIBOR+270bps交易. 一个SPV卖出10年期针对该债券名义价值为100万美元的违约掉期，该掉期的溢价为每年330bps. 同时，SPV买入100万美元

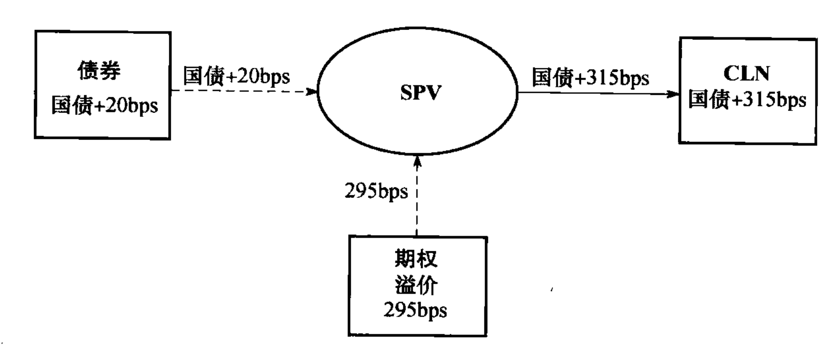


图 5.12　CLN 支付

信用评级为 AAA 的资产抵押证券，其收益为 LIBOR＋10bps. SPV 将这些现金流打包合并为一张信用联系票据，其收益为每年 LIBOR＋340bps，这样，SPV 就构建了一个合成债券，它的收益比汽车制造商发行的债券收益高出 70bps，其过程如图 5.13 所示.

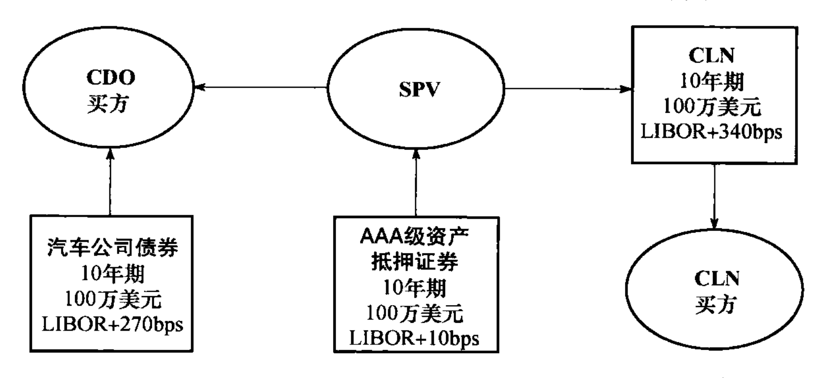


图 5.13　用 CLN 构建合成债券

如果参考资产（如汽车公司债券）发生违约，SPV 就卖出资产抵押证券来作为损失赔偿，在到期日，CLN 向票据持有人支付剩余金额. 假设回收率为 40%，那么 SPV 会以 100 万美元出售 AAA 级的资产抵押证券，向债务抵押债券买方支付 60 万美元，并将剩余的 40 万美元支付给信用联系票据持有者，如图 5.14 所示.

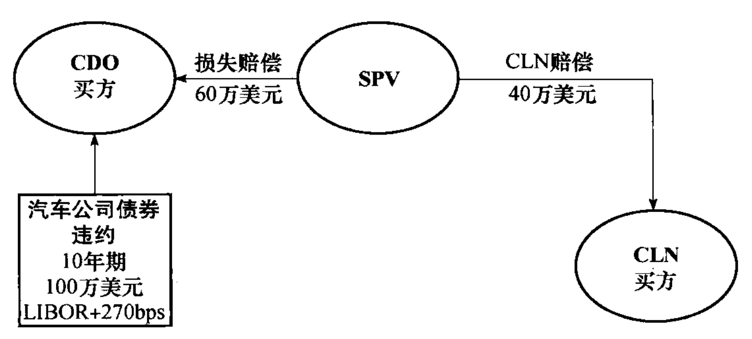


图 5.14　参考资产违约

5.12.1 含有 CLO 或 CBO 的信用联系票据

除了将信用衍生品和诸如贷款或债券的资产联系起来之外，通过对 SPV 中非投资级别的贷款和债券进行打包，信用联系票据也可以将一个违约期权和贷款抵押债券（CLO）或债券抵押债券（CBO）联系起来. 例如，机构可以将一个 SPV 中的违约银行贷款组合进行打包来创造出一种高评级的贷款抵押债券. 这个贷款抵押债券则通过另一个 SPV 和信用违约掉期联系起来，从而创造出一种信用联系票据.

为了吸引更广大的投资人，SPV 往往会将贷款/债券抵押债券和信用联系票据一样进行份额化. 图 5.15 展示了一个有优先级、次级和初级份额的特别交易机构. 每个份额都进行一个信用评级，以表示其发生违约的可能性. 通常，只有评级最高的贷款/债券抵押债券被用来构造信用联系票据. 这有助于保证信用联系票据的表现大部分能够依赖于卖出的参考资产的违约期权头寸.

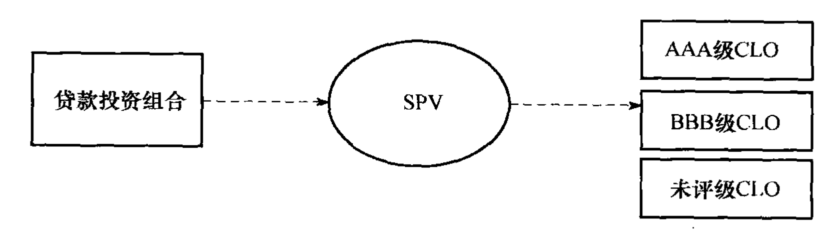


图 5.15 含有优先份额、次级份额和初级份额的 SPV

债务抵押债券包含了债券抵押债券和贷款抵押债券，这些在第 4 章中进行了详细讨论.

5.12.2 份额化信用联系票据定价

信用联系票据的价格等于每笔流入 SPV 的现金流之和，而与信用联系票据的结构无关. 但是对于一个份额化的票据而言，需要将现金流在不同份额内进行分配，从而使其总流入等于总流出. 例如，假设有一个 SPV 对于面值为 2 000 万美元的资产能够获得 LIBOR＋315bps 的收益. 该 SPV 发行双份额的信用联系票据：1 500 万美元的优先票据面值，其收益为 LIBOR；500 万美元的初级份额票据，其收益为 LIBOR 加上一个利差. 由于其中 1 500 万美元仅仅获得 LIBOR 的收益，唯一未知的现金流就是初级票据中的利差分布. 假设输入之和等于所有输出之和，利差可以通过图 5.16 的方法来计算.

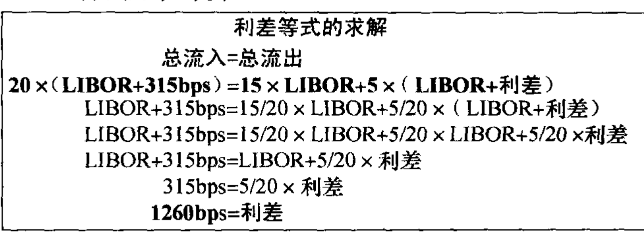


图 5.16 利差计算

5.12.3 管理资本

信用联系票据为银行提供了一个理想的减少管理资本的方法. 管理资本等于银行必须持有的权益资本和其总资产的比例，通常为 8%. 股权融资往往比较昂贵. 例如，银行股持有者往往要求年收益 15%的回报，与之相比，银行间能够以 LIBOR 进行融资. 一个减少高成本股权资本需求的方法是降低信用资产的风险. 通过从贷款组合中构造一个信用联系票据，银行能将自己承担的信用风险转移到投资人身上. 这样，只需持有 8%股权资产推荐持有比例中的 20%就可以了. 举例来说，某个银行有 50 亿美元的贷款. 该银行需要以昂贵的股权资本为其融资大约 8%，也就是 4 亿美元. 然而，如果在同样情形银行能够组成一个发行信用联系票据的 SPV，就能够将标的资产的信用风险从银行的资产负债表中去除，从而使得管理资本减少了 80%，只需 8 000 万美元.

尾注

1. O' Kane D(2001)，"Credit Derivatives Explained，" Lehman Brothers Structured Credit Research，March，pg. 3.
2. Id. pg. 3.
3. Id. pg. 3.
4. 根据 British Bankers' Association(2002)(英国行业协会 2002 年调查)，银行、券商和对冲基金占据保护买方市场的主导地位. 其中银行占总需求量的 50%. 而在卖方，则是银行和保险公司占主导.
5. Schonbucher P，*Credit Derivatives Pricing Models*，Wiley&Sons，Inc. (2003).
6. 福特公司债券的票面价值是 1 000 美元，因此总的名义价值为 1 000×10 000=1 000 万美元. 但是如果美洲银行未持有所交割债券或者持有该债券为标的物的头寸，那么就需要从市场上来购买这些债券，不过只需支付少于 1 000 万美元即可. 这是因为发生了违约的债券市场价值要小于其票面价值——价值是 400 美元. 这就给了美洲银行一个选择市场价值最低廉的债券进行交割的选项.
7. Id.，pg. 28.
8. Duffie D，and Singleton K(1999)，"Modeling Term Structures of Defaultable Bonds，" *Review of Financial Studies*，12(4)，pp. 687～720.
9. Yu Fan(2003)，"Default Correlation in Reduced-Form Models，" University of California，Irvine，working paper，September.
10. Id. pg. 1.
11. 如果 T_i 表示在违约日 τ 之前的溢价费用，那么保护买方应当支付的累计溢价部分则应该这样计算：

$$AP = sN\Delta_{i+1}\frac{\tau - T_i}{T_{i+1} - T_i} = sN\frac{T_{i+1} - T_i}{360}\frac{\tau - T_i}{T_{i+1} - T_i} = sN\frac{\tau - T_i}{360}$$

其中，s 表示名义价值或票面价值 N 的比例（点数/年计）.

12. Li D(1998)，“Constructing a Credit Curve.” Risk，Credit Risk Special Report. November，pg. 40.
13. Lando D，“On Cox Processes and Credit Risky Securities，” Department of Operations Research，University of Copenhagen(March 1998).
14. 假设 λ：$\mathcal{R}_{+}\rightarrow\mathcal{R}$并有 $\int_{0}^{\infty}\lambda(u)du=\infty$.
15. Schonbucher P. *Credit Derivatives Pricing Models*，John Wiley&Sons，Inc. (2003)，pg. 91.
16. Id. pg. 6.
17. Id. pg. 7.
18. O'Kane and Turnbull(2003)，pg. 9.
19. 假设定价日为时刻 0：T_0，因此有 $T_i-T_0=T_i$. 然而，可以用 T_v 来代替公式中的 T_0 来作为定价日，相应地换用 T_i-T_v，其中 $v=0, \cdots, i-1$.
20. O'Kane and Turnbull(2003)，pg. 8.
21. O'Kane and Turnbull(2003)，pg. 12.
22. Ibid.
23. 可以从 Bloomberg 获得历史收益率和利差数据. 进入公司债券板块中的 Yield Curve Analytics/Matrices 菜单，使用 Fair Market Yld Curves-History.
24. 年限使用表 5.1 中的 Δ_i，贴现债券收益使用美国 LIBOR.
25. 在程序中，我们将年限和利率都编入代码并存在了初始化程序 init 函数的向量中.
26. Li D(1998)，pg. 40.
27. Schonbucher P.，pg. 70.
28. 2005 年 5 月 6 日，Standard & Poor's 评级机构将通用汽车和福特公司的信用评级调整为垃圾级. 通用汽车和通用汽车接受公司——它的财务分支，都被降到了 BB 级——投资级向下两级，而福特和福特汽车信用——其财务分支被降到了 BB_{+} 级. 穆迪评级机构将福特降到了投资级中的最低级别.
29. 信用违约掉期的惯例是季度性支付，以 1/360 记，交易日顺延为下一有效交易日的记日法，每年 3 月 20 日、6 月 20 日、9 月 20 日、12 月 20 日作为到期日.
30. 可以用非线性递归法或梯度法来为风险率的校准，如 Newton-Raphson 法.
31. Kasapi A(1999)，pg. 67.
32. Id.，pg. 67.

第 6 章 天气衍生品

天气对商业企业和娱乐活动的影响巨大，并且随区域和季节变化. 很多行业，包括农业、保险业、能源业和旅游业，都受天气有利或不利的影响. 由于这个原因，金融市场创造了新型金融工具——天气衍生品，使得天气（温度）风险可以被转移或降低. 天气衍生品是关于天气指数的未定权益，天气指数是根据天气数据建立的. 普通天气指数包括但不限于日均气温(DAT)、累计年均气温（CAT)、升温天数（HDD)、降温天数（CDD)、降水、降雪和风. 与其他金融衍生品相比，天气衍生品的定价有一些困难，因为传统的风险中性无套利定价理论不管用；目前基本天气指数没有通过流动性交易工具被证券化. 在应用统计均衡定价技术方面也存在一些困难，因为观测到的天气指数是非平稳的，并且有着长期波动和趋势特征，周期要比数据所显示的长得多.[1] 与之相反，精算现值定价法相当简单并且具有直观吸引力，尽管它不能像统计模型一样把握天气的周期和趋势. 本章探索天气衍生品的各种定价模型，包括统计和随机模型的应用.

6.1 节描述天气衍生品市场的背景以及起源. 6.2 节讨论天气合约，包括芝加哥商品交易所关于天气指数的期货合约和期权合约. 6.3 节讲述建立在 Alaton 等人（2000）论文基础上的温度建模. 6.4 节讨论 6.3 节中模型的参数估计. 6.5 节讨论波动率估计，6.6 节回顾均值回归估计. 6.7 节讨论天气衍生工具的定价，包括关于 HDD 的欧式看涨和看跌期权. 6.8 节将历史日照分析作为一种定价方法. 6.9 节讨论了建立在 Campbell 和 Diebold（2005）的论文基础上的时间序列天气预测. 6.10 节讲述应用 C++进行蒙特卡罗模拟为天气期权定价.

6.1 天气衍生品市场

天气衍生品市场始于 1997 年，作为企业受厄尔尼诺现象影响的一种反应，目的是对冲会导致收益大幅下降的季节性天气变化风险. 厄尔尼诺条件与美国东部和中西部地区的暖冬相关，为消费者和企业节约了巨大的能源成本. 另外，这些条件抑制了大西洋的飓风活动，使得飓风带来的经济损失最小. 然而，同样的天气模式也与加利福尼亚的洪水相关，导致经济和生命损失.

在厄尔尼诺现象之后，天气衍生品市场迅速发展，天气衍生品合约开始在场外进行交易. 天气衍生品市场从 1997 年的基本不存在发展成 1998 年大约 5 亿美元的市场价值，但是它的流动性差，差额巨大，次级市场活动有限. 2005 年市场增长超过 50 亿美元，并且流动性更好. 天气衍生品的场外交易市场首先是由能源行业推动的. 为了扩大市场规模和规避合约交易的信用风险，芝加哥商品交易所（CME）在 1999 年 9 月创建了一个天气衍生品的电子市场. 这是可以交易标准天气衍生品的第一个交易所. 尽管天气风险对很多企业有巨大影响，包括能源生产商和消费者、连锁超市、娱乐行业和农业，但对天气衍生品的需求主要是由能源行业推动的，使得天气风险管理行业迅速发展起来.[2]

天气衍生品市场在 20 世纪 90 年代中期的发展可以归因于美国能源和公用事业行业的解除管制. 面临不断激烈的竞争和需求的不确定性，能源和公用事业企业寻求有效的套期保值工具

来稳定收益.在解除管制的环境下，能源企业家很快意识到天气状况是收入不确定的主要原因.天气不仅影响短期能源需求，还影响长期能源供给.例如，如图 6.1 所示，电力负荷量很大程度上取决于温度水平.

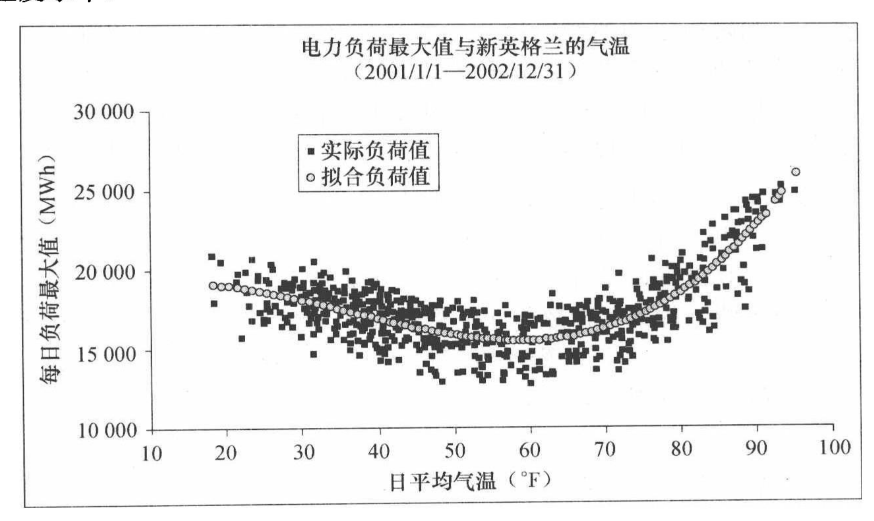


图 6.1 电力负荷最大值与新英格兰的气温
资料来源：Cao、Li 和 Wei(2004).

当日平均气温大约在 65°F时，电力负荷最大值处于最低水平，当气温上升或下降时电力负荷最大值上升.类似地，天然气消费也与气温高度相关，如图 6.2 所示.

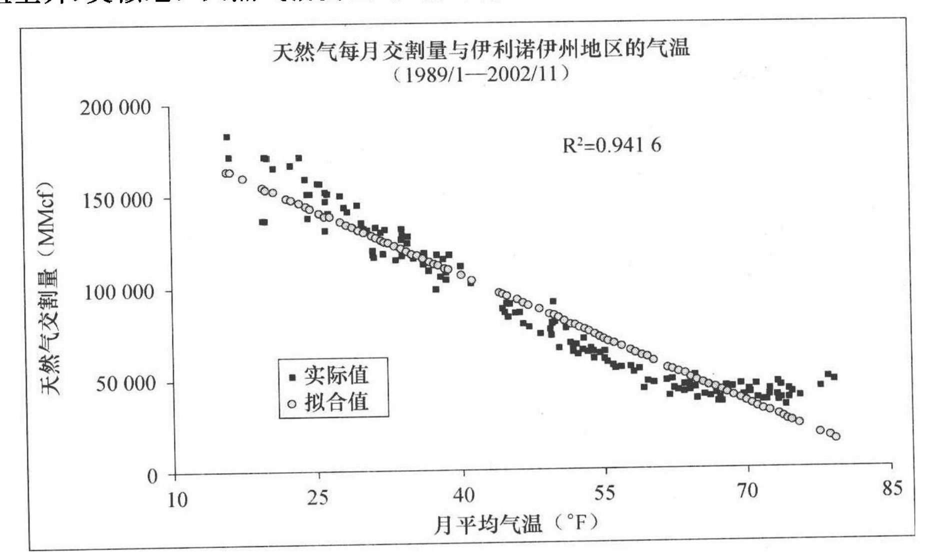


图 6.2 天然气每月交割量与伊利诺伊州地区的气温
资料来源：Cao、Li 和 Wei (2004).

因此，电力和能源（将在第 7 章中讨论）的短期需求很大程度上由天气状况驱动.另一方面，天气状况的具体模式（例如，全球变暖趋势）也会影响“能源的长期供给，因为生产商会

重新调整他们的生产水平”.[3]

Campbell 和 Diebold(2005) 提出很多有意思的理由，使得天气衍生品与其他“标准”的衍生品不同.首先，标的资产（天气）不能在现货市场上交易.其次，与金融衍生品不一样，金融衍生品在对冲价格风险上有用但不能用于对冲数量风险，而天气衍生品在对冲数量风险上有用但未必能用于对冲价格风险（尽管这两者明显相关).也就是说，天气衍生品能规避与天气相关的数量变化的风险，用期货来补充已经存在的大量商品价格风险管理工具.最后，尽管天气衍生品市场的流动性已经得到提高，但是它也不可能与传统商品市场一样好，因为天气就其性质来说是一种特定于区域的非标准的商品，而不像比如说一种特定等级的原油.合约的标的资产——天气，不能交割.因此，标准衍生品用于对冲价格风险，而天气衍生品用于对冲数量变化风险.例如，一份收费 10 000 美元的天气期权合约可能会获得每天 1 000 美元的回报，如果接下来的两个月下雨.

欧洲市场没有美国市场发展迅速，但有大量因素显示了其增长潜力.其中之一是欧洲的能源行业还没有完全解除管制，所以随着整个行业管制放松，在欧洲交易的天气衍生品数量应该会增加.这会提高市场流动性，并且鼓励新的参与者进入市场.

如果能源行业外的市场参与者对天气衍生品市场更感兴趣，也会有巨大的发展潜力.之前提到过，很多不同领域的企业都受天气影响.如果这些企业出于对冲风险的目的开始青睐天气衍生品市场，那么很可能会提高市场流动性并且增加新产品.

市场增长的另一个原因是标准合约的存在.伦敦国际金融期货交易所（LIFFE）目前正在开发跨越整个欧洲的天气期货，这会扩大整个天气衍生品市场的规模.如果市场要发展，还有一些必须移除的障碍.例如，天气数据的质量和成本在欧洲存在很大不同.想要通过历史天气数据来分析业绩的公司必须从国家气象局购买信息，这在一些国家相当昂贵.天气数据的质量要高，公司可以依赖这些数据为衍生品定价，这也很重要.

6.2 天气合约

天气衍生品通常有关于不同的标的天气指数的掉期、期货和看涨/看跌期权.普遍利用的指数有 HDD、CDD、降雨（降水）、风、水流和降雪.很多天气衍生品是关于温度天数指数的，因为这些指数使用最多.我们会集中讨论温度天气衍生品.

先介绍一下基本定义和术语.温度 T_i 是这样定义的：天气状况一定，T_i^{max}、T_i^{min} 分别表示第 i 天的最高和最低温度（以摄氏温度衡量).第 i 天的温度可以定义为：

$$T_i = \frac{T_i^{max} + T_i^{min}}{2} \tag{6.1}$$

令 T_i 表示第 i 天的温度.升温天数 HDD_i 可以定义为：

$$HDD_i = \max\{65^\circ\mathrm{F} - T_i, 0\} \tag{6.2}$$

降温天数 CDD_i 可以定义为：

$$CDD_i = \max\{T_i - 65^\circ\mathrm{F}, 0\} \tag{6.3}$$

注意，具体某一天的 HDD 和 CDD 值只是该天温度偏离参考水平的度数.在美国，将参考水平设定为 65 华氏温度（18 摄氏度）已经成为行业标准.HDD 和 CDD 的名字起源于美国能

源行业.理由是如果温度低于18℃，人们会消耗更多能源来使他们的家更温暖，而当温度高于18℃时，人们开始打开空调降温.大多数关于温度的天气衍生品是根据某一特定时期（一个日历月或者冬季/夏季）的HDD和CDD的累计值.HDD季节一般包括从11月到来年3月的冬天，CDD季节是从5月到9月.4月和10月经常被称为“肩月”.

CME的天气期货

CME提供建立在CME的温度天数（CME Degree Day）指数基础之上的标准期货合约，这一指数是某个日历月期间每天HDD和CDD的累计值，还提供关于这些期货的期权合约.CME的温度天数指数目前对11个美国城市、5个欧洲城市和2个日本城市做了特别规定.HDD/CDD指数期货是约定在将来某个特定日期购买或出售HDD/CDD指数值的合约.一份合约的名义价值等于温度天数指数乘以100美元，并且合约是以HDD/CDD指数值报价的.期货合约通过现金交割，这意味着存在以该指数为基础的每日盯市，反映为客户账户上的盈利和亏损.CME的HDD或CDD看涨期权赋予所有者以特定价格（通常称为执行或履约价格）购买一份HDD/CDD期货合约的权利，而不是义务.类似地，HDD/CDD看跌期权赋予所有者出售一份HDD/CDD期货合约的权利，而不是义务.芝加哥商品交易所的期货期权是欧式期权，这意味着只能在到期日当天执行权利.

CME期货合约将美国15个城市一个月或一个季度的HDD(CDD)的数值作为标的温度指数.时间段$[t_1, t_2]$的HDD指数被定义为连续时间变量$\int_{t_1}^{t_2}\max(65-T_t, 0)\mathrm{d}t$，CDD指数被定义为$\int_{t_1}^{t_2}\max(T_t-65, 0)\mathrm{d}t$.[4]

以一个月或一季度的累计平均气温（CAT）指数和HDD指数为标的指数的期货合约可以在5个欧洲城市交易.时间段$[t_1, t_2]$的CAT指数被定义为$\int_{t_1}^{t_2}T_t\mathrm{d}t$，其中气温是以摄氏温度而不是以华氏温度衡量.合约货币是英镑而不是美元.另外，HDD合约的温度水平设定在18℃.

对于2个日本城市（东京和大阪），期货合约以环太平洋指数为标的指数，环太平洋指数测量一个月或一季度的日平均气温.时间段$[t_1, t_2]$的环太平洋指数被定义为$\frac{1}{t_2-t_1}\int_{t_1}^{t_2}T_t\mathrm{d}t$，合约货币为日元.[5]

CME也交易不同温度指数期货的普通欧式期权.存在HDD、CDD、CAT和环太平洋指数期货的不同执行价格和到期日的看涨和看跌期权.CME的主要做市商有Aquila能源、Koch能源交易、Southern能源、Enron和Castlebridge天气市场.所有这些公司在天气衍生品的场外交易市场也很活跃.

根据Benth和Salyte-Benth(2005)，考虑以从12月到来年3月（如冬季）的一段特定时期$[t_1, t_2](t_1<t_2)$的HDD指数为标的指数的期货价格的动态波动.假定连续复利率为常数r，在时刻$t<t_1$，以HDD指数为标的指数的期货价格被定义为满足下式的$\Im_t$条件下的随机过程$F_{HDD}(\tau, t_1, t_2)$：

$$e^{-r(t_2-t)}E^Q\left[\int_{t_1}^{t_2}\max(c-T_t,0)\mathrm{d}t - F_{HDD}(\tau,t_1,t_2)\,|\,\Im_t\right]=0 \tag{6.4}$$

其中，Q 是风险中性概率，对美国或欧洲城市来说，c 等于 65℉或 18℃. 给定 $F_{HDD}(t_1, t_2)$ 的适应性，期货价格等于：

$$F_{HDD}(\tau,t_1,t_2)=E^Q\left[\int_{t_1}^{t_2}\max(c-T_t,0)\mathrm{d}t\,|\,\Im_t\right] \tag{6.5}$$

类似地，CDD 期货价格等于：

$$F_{CDD}(\tau,t_1,t_2)=E^Q\left[\int_{t_1}^{t_2}\max(T_t-c,0)\mathrm{d}t\,|\,\Im_t\right] \tag{6.6}$$

同理，CAT 期货和环太平洋指数期货的价格为：

$$F_{CAT}(\tau,t_1,t_2)=E^Q\left[\int_{t_1}^{t_2}T_t\mathrm{d}t\,|\,\Im_t\right] \tag{6.7}$$

和

$$F_{PRIM}(\tau,t_1,t_2)=E^Q\left[\frac{1}{t_2-t_1}\int_{t_1}^{t_2}T_t\mathrm{d}t\,|\,\Im_t\right] \tag{6.8}$$

注意，

$$F_{PRIM}(\tau,t_1,t_2)=\frac{1}{t_2-t_1}F_{CAT}(\tau,t_1,t_2) \tag{6.9}$$

另外，由于 $\max(c-x,\ 0)=c-x+\max(x-c,\ 0)$，可得：

$$F_{HDD}(\tau,t_1,t_2)=c(t_2-t_1)-F_{CAT}(\tau,t_1,t_2)+F_{CDD}(\tau,t_1,t_2)$$

在芝加哥商品交易所外，有大量不同的合约在场外市场交易. 例如，HDD 看涨期权的买方在合约开始日付给期权卖方一笔费用. 如果合约期间 HDD 的数值大于事先确定的水平，作为回报，期权买方能获得收益. 收益的多少由执行水平和最小变动单位决定. 最小变动单位是指合约期内升温天数每高于执行水平一天，看涨期权的买方可以获得的金额. 期权通常对最大支付额有一定限制，不像传统的股票期权. 通过规定以下参数可以制定一份标准的天气期权：

- 合约类型（看涨或看跌）.
- 合约期限（如一月份）.
- 标的指数（HDD 或 CDD）.
- 获得气温数据的官方气象站.
- 执行水平.
- 最小变动单位.
- 最大收益（如果存在）.

为找到期权收益的公式，设定 K 表示合约规定的执行水平和最小变动单位，合约期限为 n 天. 因此，合约期限内的 HDD 和 CDD 的数值分别是：

$$H_n=\sum_{i=1}^{n}HDD_i \text{ 和 } C_n=\sum_{i=1}^{n}CDD_i \tag{6.10}$$

现在我们可以得到一份不设限制的 HDD 看涨期权的价值：

$$\alpha\max\{H_n-K,0\} \tag{6.11}$$

与 HDD 看跌期权和 CDD 看涨/看跌期权类似的期权价值可以用同样的方式确定. 图 6.3

给出了基于 HDD 与 CDD 的远期合约和期权合约的例子.

基于 HDD 与 CDD 的远期合约和期权合约

	HDD 远期合约	CDD 看跌期权
当前时间	2001－12－01	2002－01－01
所在地	Phil，Int'l Airport，Philadelphia	Hartsfield Airport，Atlanta
多头头寸	ABC 银行	空调有限公司
空头头寸	电力供应有限公司	XYZ 银行
累计日期	2002/02	2002/07
最小变动单位	每 HDD 84 000	每 CDD 10 000 美元
执行水平	855 HDD	550 CDD
实际水平	650 HDD	510 CDD
到期日收益		
（多头）	(650－850)×4 000＝－800 000 美元	(550－510)×10 000＝400 000 美元

图 6.3

资料来源：Cao、Li 和 Wei (2004).

6.3 温度建模

要对天气衍生品建模，我们必须推导出标的变量，即温度的随机过程，并了解其行为和运动. 温度具有很强的季节性变化和模式. 因此，温度不可能用随机游走很好地模拟. 同时还有其他观测量需要考虑：温度表现出很高的自相关性，这意味着短期行为和长期行为是不同的. 最后，因为温度（或降水）不能买卖，所以不存在标的资产. 没有办法通过构建金融资产组合来复制天气衍生品的价值. 考虑到上述问题，Black-Scholes 框架无法应用. 图 6.4 指出了从 1994 年 1 月 1 日到 2005 年 1 月 1 日期间瑞典斯德哥尔摩日平均气温的典型模式.

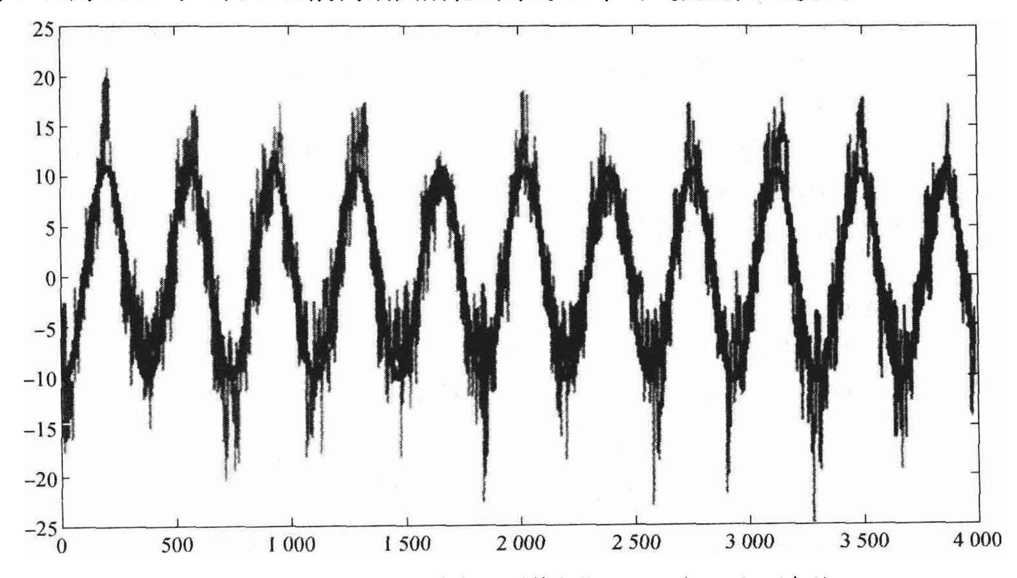


图 6.4 瑞典斯德哥尔摩的日平均气温（2004/01—2005/01）

资料来源：Alaton P、Djehiche B 和 Sillberger D(2000).

考虑到温度的季节性和周期性，模型应该将均值回归包含在过程中. 平均温度似乎在夏季的 20℃左右和冬季的－10℃左右变化. 快速浏览图 6.1 后，我们猜测采用正弦函数来描述季节性是可能的. 函数形式如下：

$$\sin(\omega t+\theta)$$

其中，t 表示时间，以天计算. 令 $t=1$，2，…表示 1 月 1 日，1 月 2 日等. 因为我们知道震荡周期为一年（忽视闰年），可得 $\omega=\frac{2\pi}{365}$. 因为一年中最小和最大的平均温度不常发生在 1 月 1 日和 7 月 1 日，所以我们不得不引出一个滞后角度 θ. 此外，对数据序列更深入的观察揭示了数据具有一个正向趋势. 趋势很弱，但确实存在. 平均温度确实逐年在增长. 对此有非常多的原因. 一个原因是世界范围内的全球变暖趋势. 另一个原因被称作城市热效应，指大城市附近地区的温度趋向于上升，因为大城市在发展，环境在变暖. 为描述数据的这种弱趋势，我们假设上升趋势是近似线性的. 我们本可以假设是多项式的，但由于其对平均温度整体动态的弱效应，只有此多项式中的线性项占主导作用.

总结下来，t 时刻平均气温 T_t^m 的决定性模型具有如下形式：

$$T_t^m=A+Bt+C\sin(\omega t+\theta) \tag{6.12}$$

其中，必须选择参数 A，B，C，θ 的值以使曲线能很好地拟合数据.

由于受全球和季节性天气变化的影响，气温的波动不能预先决定. 因此，为得到更符合实际的模型，必须在模型（6.12）中加入一些噪声项. 一个选择是标准的维纳（Wiener）过程 $(W_t,\ t\geqslant 0)$. 事实上，这样的选择不仅仅对于模型的数学可延续性是合理的，同时图 6.5 也表

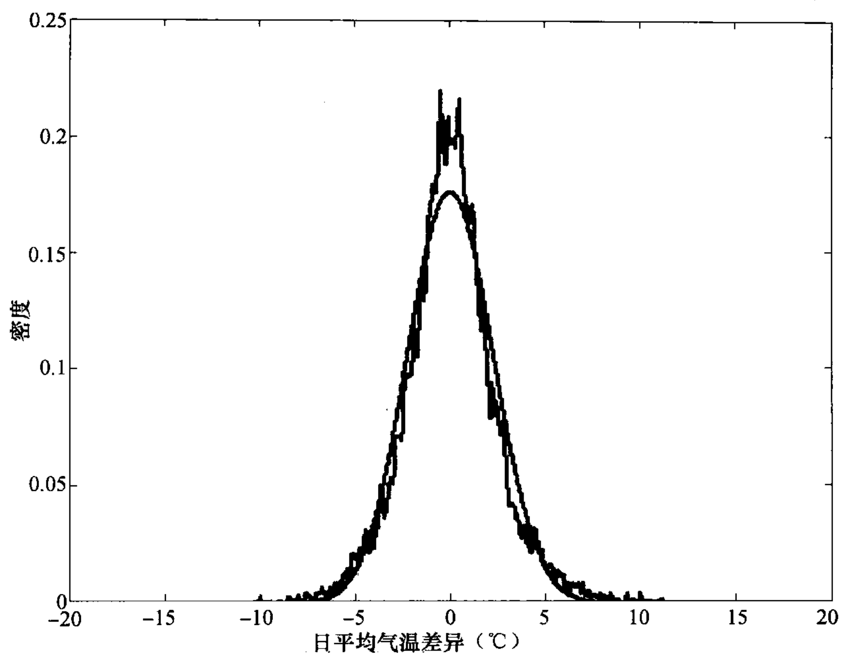


图 6.5　日平均气温分布

资料来源：Alaton P、Djehiche B 和 Sillberger D(2000)，9.

明每日温差和对应的正态分布能够很好地拟合，尽管每日平均温度中产生微小差异的概率将会被略微低估.

6.3.1 噪声过程

对数据系列更深入的观察，揭示了温度方差 σ_t^2 在一年的不同月份中都是不断变化的，但在每个月内近似为常数. 特别是在冬天，二次方差比一年其他时间高很多. 因此，我们假设 σ_t 是一个分段常数函数，每一月份都有一个常数值.[6] 可以将 σ_t 写成：

$$\sigma_t = \begin{cases} \sigma_1 & 1\text{ 月} \\ \sigma_2 & 2\text{ 月} \\ \vdots & \\ \sigma_n & 12\text{ 月} \end{cases}$$

其中，$\sigma_i(i=1, \cdots, 12)$ 是正常数. 因此，温度的驱动噪声过程为 $(\sigma_t W_t, t \geqslant 0)$.

6.3.2 均值回归

我们知道，温度不能在长时间内不断上升. 这意味着我们的模型不能允许温度偏离其均值超过一段时间. 换句话说，我们所寻找的描述温度的随机过程应该具有均值回归特性. 将所有假设放在一起，通过以下随机微分方程的随机过程解为温度建模：[7]

$$dT_t = a(T_t^m - T_t)dt + \sigma_t dW_t \tag{6.13}$$

其中，$a \in \mathbb{R}$ 决定了均值回归速度. 这个方程的解通常被称作 Ornstein-Uhlenbeck 过程. Alaton 等人（2000）建议加上另外一个漂移项，因为方程（6.13）在长期并不是真的均值回归到 T_t^m，参见 Dornier 和 Queruel（2000）的证明：[8]

$$\frac{dT_t^m}{dt} = B + \omega C\cos(\omega t + \theta) \tag{6.14}$$

假设平均温度 T_t^m 不是常数，该项将调整漂移项使 SDE 的解具有长期均值 T_t^m.

从 $T_s = x$ 出发，得到下列温度模型：

$$dT_t = \left(\frac{dT_t^m}{dt} + a(T_t^m - T_t)\right)dt + \sigma_t dW_t \quad t > s \tag{6.15}$$

其解为：

$$T_t = (x - T_s^m)e^{-a(t-s)} + T_t^m + \int_s^t e^{-a(t-\tau)}\sigma_\tau dW_\tau \tag{6.16}$$

其中，

$$T_t^m = A + Bt + C\sin(\omega t + \theta) \tag{6.17}$$

6.4 参数估计

为估计未知参数 A、B、C、θ、a 和 σ，我们可以运用最小二乘法回归技术. 为求出公式（6.17）中的参数值，我们可以用最小二乘法将函数（6.18）和温度数据进行拟合，从而估计公式（6.18）的参数值：

$$Y_t = a_1 + a_2 t + a_3 \sin(\omega t) + a_4 \cos(\omega t) \tag{6.18}$$

我们必须找到参数向量 $\xi=(a_1, a_2, a_3, a_4)$，使得：

$$\min_{\xi} \| \boldsymbol{Y}-\boldsymbol{X} \|^2 \tag{6.19}$$

其中，$\boldsymbol{Y}$ 是公式（6.18）中的元素向量，$\boldsymbol{X}$ 是（历史）数据向量. 从而模型（6.17）中的常数为：

$$A = a_1 \tag{6.20}$$

$$B = a_2 \tag{6.21}$$

$$C = \sqrt{a_3^2 + a_4^2} \tag{6.22}$$

$$\theta = \arctan\left(\frac{a_4}{a_3}\right) - \pi \tag{6.23}$$

作为一个例子，Alaton 等人（2000）利用过去 40 年瑞典斯德哥尔摩 Bromma 机场的温度数据估计公式（6.18）中的参数，得到平均温度函数如下：

$$T_t^m = 5.97 + 6.57 \cdot 10^{-5} t + 10.4 \sin\left(\frac{2\pi}{365} t - 2.01\right) \tag{6.24}$$

正弦函数的振幅大约为 10℃，这意味着冬天和夏天的温度差异大约为 20℃.[9]

6.5 波动率估计

Alaton 等人（2000）提供了从每月收集的数据中估计 σ 的估计值. 给定一个 N_i 天的具体月份 μ，用 $T_j(j=1, \cdots, N_i)$ 表示月份 μ 观测的温度值. 第一个估计值是基于 T_t 的二次方差（参见 Basawa 和 Prasaka Rao[1980]，212～213 页）.[10]

$$\hat{\sigma}_i^2 = \frac{1}{N} \sum_{j=0}^{N_\mu - 1} (T_{j+1} - T_j)^2 \tag{6.25}$$

第二个估计值是将公式（6.15）离散化，并将离散化得到的方程看做回归方程得到的. 在给定月份 μ 期间，离散方程为：

$$T_j = T_j^m - T_{j-1}^m + a T_{j-1}^m + (1-a) T_{j-1} + \sigma_\mu \varepsilon_{j-1}, \quad j = 1, \cdots, N_\mu \tag{6.26}$$

当 $\{\varepsilon_j\}_{j=1}^{N_m - 1}$ 为独立标准正态分布随机变量，如 $\varepsilon_j \sim N(0, 1)$. 令 $\widetilde{T}_j = T_j - (T_j^m - T_{j-1}^m)$，可以将（6.26）写为：

$$\widetilde{T}_j = a T_{j-1}^m + (1-a) T_{j-1} + \sigma_\mu \varepsilon_{j-1} \tag{6.27}$$

该式可以看做由昨日的气温回归出今天的气温. 因此，σ_i 的有效估计为（参见 Brockwell 和 Davis[1990]）：[11]

$$\hat{\sigma}_\mu^2 = \frac{1}{N_\mu - 2} \sum_{j=0}^{N_\mu} (\widetilde{T}_j - a T_{j-1}^m + (1-a) T_{j-1})^2 \tag{6.28}$$

为了评价（6.28）中的估计值，需要找出均值回复的参数 α 的估计值. 这将在下一节讨论.

6.6 均值回复参数估计

利用 Bibby 和 Sorensen(1995) 给出的鞅估计函数方法可以估计出**均值回复**（mean reversion）的参数.[12] 令 $b(T_t; a)$ 表示公式（6.15）中天气随机过程的漂移项：

$$b(T_t;a)=\frac{\mathrm{d}T_t^m}{\mathrm{d}t}+a(T_t^m-T_t) \tag{6.29}$$

根据 Alaton 等人（2000）的研究，基于 n 天内累积的观测值，通过等于零的方程可以获得 a 的有效估计值$\hat{a}_n$：

$$G_n(\hat{a}_n)=0 \tag{6.30}$$

其中，

$$G_n(a)=\sum_{i=1}^{n}\frac{\dot{b}(T_{i-1};a)}{\sigma_{i-1}^2}(T_i-E[T_i\mid T_{i-1}]) \tag{6.31}$$

$b(T_t;a)$ 表示公式（6.29）中漂移项关于 a 的导数. 为了求解方程（6.30），我们仅仅需要确定公式（6.31）中的每一个条件期望项 $E[T_i\mid T_{i-1}]$. 当 $t\geqslant s$ 时，通过公式（6.16），

$$T_t=(T_s-T_s^m)\mathrm{e}^{-a(t-s)}+T_t^m+\int_s^t\mathrm{e}^{-a(t-\tau)}\sigma_\tau\mathrm{d}W\tau \tag{6.32}$$

可推出：

$$E[T_i\mid T_{i-1}]=(T_{i-1}-T_{i-1}^m)\mathrm{e}^{-a}+T_i^m \tag{6.33}$$

与原先一样：

$$T_t^m=A+Bt+C\sin(\omega t+\theta)$$

因此，

$$G_n(a)=\sum_{i=1}^{n}\frac{T_{i-1}^m-T_{i-1}}{\sigma_{i-1}^2}(T_i-(T_{i-1}-T_{i-1}^m)\mathrm{e}^{-a}-T_i^m) \tag{6.34}$$

从中可推出均值回归估计值：

$$\hat{a}_n=-\log\left(\frac{\sum_{i=1}^{n}Y_{i-1}(T_i-T_i^m)}{\sum_{i=1}^{n}Y_{i-1}(T_{i-1}-T_{i-1}^m)}\right) \tag{6.35}$$

是方程（6.30）的唯一零值，其中，

$$Y_{i-1}=\frac{T_{i-1}^m-T_{i-1}}{\sigma_{i-1}^2},\quad i=1,2,\cdots,n \tag{6.36}$$

6.7 天气衍生品的定价

6.7.1 模型框架

对于定价模型，定义如下参数：

- T_t^m：时刻 t 模型化后平均温度.
- T_t：时刻 t 的当前温度.
- a：均值回归参数.
- σ：温度的波动率.
- W_t：维纳过程.

天气衍生品市场是不完全市场的一个经典例子，因为标的变量温度是不可贸易的. 因此，

为了获得这些合约的唯一价格，我们不得不考虑风险 λ 的市场价格. 因为现在还不存在一个真实的市场可以获得价格，为了简化，假定风险的市场价格是不变的. 进一步说，假设给定一个拥有不变利率 r 的无风险资产，并且对于每一个摄氏度支付一个货币单位. 因此，在鞅测度 Q 下，温度过程具有风险 λ 的市场价格特征，同样可由满足如下动力学的 T_t 表示：

$$\mathrm{d}T_t = \left(\frac{\mathrm{d}T_t^m}{\mathrm{d}t} + a(T_t^m - T_t) - \lambda\sigma_t\right)\mathrm{d}t + \sigma_t \mathrm{d}W_t \tag{6.37}$$

其中 $\{W_t, t\geqslant 0\}$ 是一个 Q 维纳过程. 根据 Alaton 等人（2000）的研究，我们开始计算 T_t 的期望值和方差，因为衍生品的价格由风险中性鞅测度 Q 的贴现期望值表示. 我们将在物理测度 P 下使用 Girsanov 定理来改变漂移项. 然而，因为 Girsanov 变换仅仅改变了漂移项，所以在两种测度下，T_t 的方差是一样的. 因此，

$$\mathrm{Var}[T_t \mid \Im_s] = \int_s^t \sigma_u^2 \mathrm{e}^{-2a(t-u)} \mathrm{d}u \tag{6.38}$$

进一步来说，它来自公式（6.16）：

$$E^P[T_t \mid \Im_s] = (T_s - T_s^m)\mathrm{e}^{-a(t-s)} + T_t^m. \tag{6.39}$$

因此，鉴于公式（6.37），必须有如下式子：

$$E^Q[T_t \mid \Im_s] = E^P[T_t \mid \Im_s] - \int_s^t \lambda\sigma_u \mathrm{e}^{-a(t-u)} \tag{6.40}$$

估计其中的积分项，这里 σ 是常数，得到：

$$E^Q[T_t \mid \Im_s] = E^P[T_t \mid \Im_s] - \frac{\lambda\sigma_i}{a}(1 - \mathrm{e}^{-a(t-s)}) \tag{6.41}$$

并且方差为：

$$\mathrm{Var}[T_t \mid \Im_s] = \frac{\sigma^2}{2a}(1 - \mathrm{e}^{-2a(t-s)}) \tag{6.42}$$

对于 $0\leqslant s\leqslant t\leqslant u$，两个不同日期的温度的协方差是：

$$\mathrm{Cov}[T_t, T_u \mid \Im_s] = \mathrm{e}^{-a(u-t)} \mathrm{Var}[T_t \mid \Im_s] \tag{6.43}$$

Alaton 等人（2000）的研究表明，现在假定 t_1 和 t_n 表示一个月的第一天和最后一天，并且从 $[t_1, t_n]$ 的某一个月份时点 s 开始过程. 为了计算这个案例中 T_t 的期望值和方差，我们将公式（6.40）和公式（6.38）中的积分拆分为两个部分，其中每一个部分 σ 都是常数. 接着得到：

$$E^Q[T_t \mid \Im_s] = E^P[T_t \mid \Im_s] - \frac{\lambda}{a}(\sigma_i - \sigma_j)\mathrm{e}^{-a(t-t_1)} + \frac{\lambda\sigma_i}{a}\mathrm{e}^{-a(t-s)} - \frac{\lambda\sigma_j}{a} \tag{6.44}$$

方差为：

$$\mathrm{Var}[T_t \mid \Im_s] = \frac{1}{2a}(\sigma_i^2 - \sigma_j^2)\mathrm{e}^{-2a(t-t_1)} - \frac{\sigma_i^2}{2a}\mathrm{e}^{-2a(t-s)} + \frac{\sigma_j^2}{2a} \tag{6.45}$$

现在可以将其直接推广到较大的时间间隔.

6.7.2 定价加温日（HDD）期权

为了对标准的 HDD 看涨期权定价，我们利用公式（6.5）中给定的收益：

$$\chi = \alpha\max(H_n - K, 0) \tag{6.46}$$

其中，

$$H_n = \sum_{i=1}^{n} \max(65°\mathrm{F} - T_t, 0) \tag{6.47}$$

收益取决于某时间区间 HDD 的累积（例如，一月份的冬季）.

为了简化起见，假定最小价位大小是 $\alpha=1$. 式（6.46）中的合约属于算术平均亚洲期权类. 对于对数正态分布标的过程情况，这样一个期权价格的确切方程是未知的，所以我们试图做一些近似处理. 我们知道在风险中性测度 Q 下，给定时刻 s 的信息，

$$T_t \sim N(\mu_t, \sigma_t^2)$$

其中，μ_t 由公式（6.44）给定，σ_t^2 由公式（6.45）给定. 假定在冬季 $\max(65-T_t)=0$ 的概率是非常小的. 那么，对于这样一个合约，我们可以写出：

$$H_n = 65n - \sum_{i=1}^{n} T_{t_i} \tag{6.48}$$

我们现在确定在（正态）分布模型下的合理定价. 根据 Alaton 等人（2000）的研究，我们知道 $T_{t_i}(i=1, \cdots, n)$ 是来自 Ornstein Uhlenbeck 过程的所有样本，它是一个高斯过程. 这意味着向量（T_{t_1}，T_{t_2}，…，T_{t_n}）是高斯的. 因为公式（6.48）之和是向量元素的线性组合，H_n 同样是高斯的.[13] 我们计算一阶矩和二阶矩. 对于 $t<t_1$，得到：

$$E^Q[H_n | \Im_s] = E^Q\left[65n - \sum_{i=1}^{n} T_{t_i} | \Im_s\right] = 65n - \sum_{i=1}^{n} E^Q[T_{t_i} | \Im_t] \tag{6.49}$$

和

$$\mathrm{Var}[H_n | \Im_s] = \sum_{i=1}^{n} \mathrm{Var}[T_{t_i} | \Im_s] + 2\sum_{j=1}^{n} \sum_{i<j} \mathrm{Cov}[T_{t_i}, T_{t_j} | \Im_t]. \tag{6.50}$$

假定对先前公式进行计算，并且求得：

$$E^Q[H_n | \Im_t] = \mu_n \text{ 和 } \mathrm{Var}[H_n | \Im_t] = \sigma_n^2 \tag{6.51}$$

那么，$H_n \sim N(\mu_n, \sigma_n)$. 因此，时刻 $t \leqslant t_1$ 收益（6.46）的价格给定为：

$$\begin{aligned} c_{HDD}(t) &= \mathrm{e}^{-r(t_n-t)} E^Q[\max(H_n - K, 0) | \Im_t] \\ &= \mathrm{e}^{-r(t_n-t)} \int_K^{\infty} (x-K) f_{H_n}(x) \mathrm{d}k \\ &= \mathrm{e}^{-r(t_n-t)} \left((\mu_n - K)\Phi(-\alpha_n) + \frac{\sigma_n}{\sqrt{2\pi}} \mathrm{e}^{-\frac{\alpha_n^2}{2}} \right) \end{aligned} \tag{6.52}$$

其中，$\alpha_n = (K-\mu_n)/\sigma_n$ 和 Φ 表示累积标准正态分布函数（参见 Platen 和 West[2003]，23 页[14]）.

类似地，我们衍生出 HDD 看跌期权价格的公式，它的应得收益为：

$$y = \max(K - H_n, 0) \tag{6.53}$$

价格为：

$$\begin{aligned} p_{HDD}(t) &= \mathrm{e}^{-r(t_n-t)} E^Q[\max(K - H_n, 0) | \Im_t] \\ &= \mathrm{e}^{-r(t_n-t)} \int_0^K (K-x) f_{H_n}(x) \mathrm{d}x \\ &= \mathrm{e}^{-r(t_n-t)} \left\{ (K-\mu_n)\left(\Phi(\alpha_n) - \Phi\left(-\frac{\mu_n}{\sigma_n}\right)\right) + \frac{\sigma_n}{\sqrt{2\pi}}\left(\mathrm{e}^{-\frac{\alpha_n^2}{2}} - \mathrm{e}^{-\frac{1}{2}\left(\frac{\mu_n}{\sigma_n}\right)^2}\right)\right\} \end{aligned} \tag{6.54}$$

对于看涨期权和看跌期权，式（6.52）和式（6.54）各自主要在冬季月份持有合约，冬季月份主要是12月份到3月份.[15]在夏季月份，我们不能在没有约束的条件下使用这些公式.如果平均气温非常接近于或者是高于65°F(18℃)，则不再有 $\max(65-T_{t_i}, 0)\neq 0$. 对于这样的合约，我们可以使用蒙特卡罗模拟.

Zeng(2000) 讨论了天气衍生品的纯概率统计方法（Platen 和 West[2004]），同样考虑了基于预测的定价方法.在后一方法中，通过假定期权 CDD 遵循均值和标准差分别等于历史数据的样本均值和标准差的正态分布，正态分布适合于历史 CDD 数据. "非超越"概率平均划分为最高、中间、最低的三部分，如图 6.6 所示.

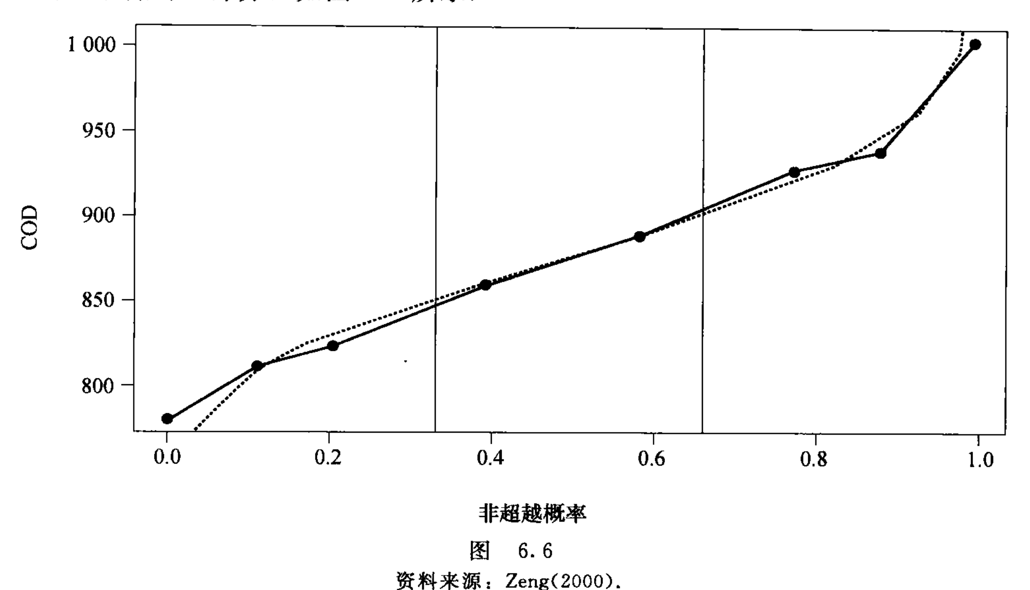


图 6.6

资料来源：Zeng(2000).

接着对拟合分布取样，最高、中间、最低的三部分样本数目与高于、接近、低于气候基准（各自用 p_A、p_B、p_C 表示）成比例.例如，与样本拟合分布均匀地并行于概率分布的传统蒙特卡罗方法不一样，基于预测的方法仅仅从来自最高、中间、最低三部分的样本拟合分布，这样可以合并概率气候预测到样本 CDD 值（一种有偏样本蒙特卡罗方法）.就像传统蒙特卡罗方法一样，这个样本分布的收益被平均化.该方法认为当天气有季节特性时（例如，较高的7月气温与较高的6、7、8月（JJA）气温是相关的），历史数据的等级排序相关会很高，因此温度 p_A、p_B、p_C 的预测异象概率被认为分别近似于 CDD 高于、接近和低于气候基准的概率.[16]

6.8 历史日照分析

历史日照分析方法利用历史数据评估合约，并用已实现收益的均值作为合理价值的估计.例如，假定一个看涨期权是7月城市的 CDD，并假定我们有20年的日度气温.为了应用历史日照分析方法，我们利用过去20年的每一个7月已实现的 CDD 计算期权收益.20个收益的平均值是看涨期权价值的估计.因此，这个方法的关键假设是过去收益的分布精确地描述了未来收益的分布.在大多数案例中，这是一个意义深远的要求.例如，在前面的案例中，我们仅有20个收益的观测，这很难捕捉到真实分布的完全特性.

Cao、Li 和 Wei(2004) 应用这个方法到亚特兰大、芝加哥和纽约三个月（1 月、2 月和 3 月）累积 HDD 看涨期权. 表 6.1 给出了计算过程. 他们首先计算了每一年实现的累积 HDD，接着评估了相应的期权收益.[17]（对于亚特兰大、芝加哥和纽约，执行价格分别被设定为 1 500、3 200和 2 500.）例如，当 Cao、Li 和 Wei(2004) 使用亚特兰大所有 20 个历史观测值时，平均收益为 92.15；当最近的 19 个观测值被使用时，平均收益是 82.37，等等. 当他们仅仅使用最近的 10 个观测值时，合理价值的估计是 22.5. 回溯过去 10 年与回溯过去 20 年将导致价值估计超过 300%的偏差！纽约 HDD 期权的估计有最小的偏差. 但即使在那里，最高估计超过最低估计 70%.

表 6.1

	HDDs for Jan. 1-March 31			HDD 看涨期权收益			年	看涨期权价值的估计		
年	亚特兰大	芝加哥	纽约	亚特兰大	芝加哥	纽约	平均	亚特兰大	芝加哥	纽约
1979	1 778	3 851	2 841	278	651	341				
1980	1 672	3 474	2 702	172	274	202				
1981	1 698	3 183	2 708	198	0	208				
1982	1 587	3 760	2 791	87	560	291				
1983	1 749	3 002	2 443	249	0	0				
1984	1 660	3 424	2 724	160	224	224				
1985	1 723	3 591	2 567	223	391	67				
1986	1 416	3 208	2 533	0	8	33				
1987	1 602	2 813	2 504	102	0	4				
1988	1 649	3 417	2 593	149	217	93	10	22.50	63.70	69.60
1989	1 242	3 150	2 417	0	0	0	11	34.00	77.64	71.73
1990	1 009	2 627	2 078	0	0	0	12	39.67	71.17	66.08
1991	1 354	3 066	2 217	0	0	0	13	36.62	66.31	63.54
1992	1 325	2 862	2 439	0	0	0	14	49.93	89.50	63.79
1993	1 514	3 277	2 667	14	77	167	15	57.27	98.47	74.47
1994	1 410	3 524	2 921	0	324	421	16	69.25	92.31	69.81
1995	1 295	3 096	2 370	0	0	0	17	70.29	119.82	82.82
1996	1 666	3 410	2 608	166	210	108	18	77.39	113.17	89.78
1997	1 102	3 226	2 377	0	26	0	19	82.37	121.63	95.68
1998	1 545	2 637	2 060	45	0	0	20	92.15	148.10	107.95
							最高	92.15	148.10	107.95
							最低	22.50	63.70	63.54
							最高/最低	4.10	2.32	1.70

资料来源：Cao、Li 和 Wei(2004).

我们可以认为需要使用尽可能长的时间序列来增加精确度. 然而，使用更多的数据将覆盖

更多的温度变化，但一个衍生品证券收益取决于未来的温度行为，而未来的气温与过去是大不相同的．当衍生品证券的成熟期很短时，这尤其正确．最终，决定归结为统计有效性和代表性之间的权衡．[18]行业内被普遍接受的样本长度为 20～30 年．进一步来说，我们可以结合温度预测的日照分析来获得更具代表性的价格估计．

同保险或精算模型类似，分析历史日照现象的方法并不能核算和温度变量相关的风险的市场价格．[19]这些方法仅仅适用于单一交易者的视角．[20]为了确定包含风险溢价的市场价格，建立如下前向模型：

$$dY(t) = \beta[\theta(t) - Y(t)]dt + \sigma(t)Y_t^r dW(t) \tag{6.55}$$

其中，$Y(t)$ 是当前温度，$\theta(t)$ 是确定的长期温度水平，β 是瞬间温度转化为长期稳定水平 $\theta(t)$的速率．$\sigma(t)$ 是指振幅，它与季节相关．$r=0$，0.5 或 1．$W(t)$ 是一个维纳随机过程，用来衡量温度的随机更新．为了估计参数 β 和 $\theta(t)$、$\sigma(t)$ 中包含的参数，我们将公式（6.55）离散化为：

$$Y_t - Y_{t-1} = \beta[\theta(t) - Y_t]\Delta t + \sigma(t)Y_t^r \Delta W_t$$

其中，函数 $\theta(t)$ 和 $\sigma(t)$ 经过细致的统计分析是可以确定的．一旦通过最小二乘法确定了公式（6.55），那么其他的未定权益都可以通过取贴现未来收益的期望值来确定，即

$$X = e^{-r(T-t)}E[g(Y_t, Y_{t+1}, \cdots, Y_T)] \tag{6.56}$$

其中，X 是未定权益的现值，r 是无风险利率，T 是权益到期日，$g(Y_t,\ Y_{t+1},\ \cdots,\ Y_T)$ 是时刻 T 时的回报，通常是已获取温度（Y_t，Y_{t+1}，$\cdots$，Y_T）的函数（例如，累积 HDD 或 CDD 合约）．

给定 $\theta(t)$ 和 $\sigma(t)$ 的复杂形式和大部分收益的路径相关性，公式（6.55）通常没有闭路解，必须使用蒙特卡罗模拟．这种持续的调整存在两点不足：第一，没有考虑到市场价格的风险，那么假定的风险中性估值并没有经过任何理论上的证明；第二，公式（6.55）并没有反映出日常温度间存在的固有的序列相关性．[21]由于存在以上两点瑕疵，包含序列相关性的时间序列离散随机分布和模型已用来预测温度，详见下节的讨论（参考 Campbell 和 Diebold[2002]；Cao 和 Wei[2003]）．

6.9 时间序列天气预测

如图 6.4 所示，某市日均温度的高温期间（夏天）和低温期间（冬天）是有重复并且有规律的，但是不同城市间的季节性的波动在振幅和波动类型上都有显著不同．[22]但是，大多数城市的绝对温度分布都是双峰的，峰值分别出现在最低温和最高温．[23]不同城市的气温时间序列表明季节项将成为拟合日平均气温任何时间序列模型的一个重要因素，因为“平均气温展现了季节变化，季节项的振幅和精确性在各个城市显著不同”．[24]Campbell 和 Diebold(2000) 基于两方面原因使用一个低阶傅里叶级数替换日度虚拟变量．第一，使用低阶傅里叶级生成一个平滑季节项，平滑季节项来自于基本直觉“不同季节的过渡是渐进的，而不是不连续的．”[25]第二，傅里叶近似“会导致将要估计的参数大幅降低，这种降低显著减少了计算时间，增加了计算稳定性．”[26]

尽管季节性因素占主导，Campbell 和 Diebold(2000) 同样加入了非季节项，这在日度平均气温的动态性中也许是有效的；特别是，一个确定的线性趋势和周期——持久性（但协方差平稳性）与趋势和季节性分离.[27]这样的周期动态性可以使用自回归滞后项捕捉，而自回归滞后项是方便数值上稳定参数估计的.因此，用自回归模型预测和估计未来平均气温表明：

$$T_t = Trend_t + Seasonal_t + \sum_{l=1}^{L} \rho_{t-1} T_{t-1} + \sigma \varepsilon_t \tag{6.57}$$

其中，

$$Trend_t = \beta_0 + \beta_1 t$$

和

$$Seasonal_t = \sum_{p=1}^{P} \left(\delta_{c,p} \cos\left(2\pi p \frac{d(t)}{365} \right) + \delta_{s,p} \sin\left(2\pi p \frac{d(t)}{365} \right) \right) \tag{6.58}$$

$$\varepsilon_t \overset{iid}{\sim} N(0,1)$$

其中，$d(t)$ 是周期为 1，…，365 的重复跃阶函数（例如，一年的每一天假定为 1～365 的一个值）.模型 1 使用普通最小二乘估计得出，它对平均气温的不变项、趋势项、傅里叶项和滞后平均气温项回归，使用的是 $L=25$ 的自回归项（为了捕捉长期记忆动力）和傅里叶正弦与余弦项（$P=3$）.

根据方差残差的相关曲线图，Campbell 和 Diebold(2002) 发现公式（6.57）中的模型有条件异方差，尽管残差自相关是可忽略的，并且与白噪声是一致的，模型设定明显与非线性依赖性相关.因此，Campbell 和 Diebold(2002) 修改模型为：

$$T_t = Trend_t + Seasonal_t + \sum_{l=1}^{L} \rho_{t-1} T_{t-l} + \sigma_t \varepsilon_t \tag{6.59}$$

其中，

$$Trend_t = \beta_0 + \beta_1 t$$

和

$$Seasonal_t = \sum_{p=1}^{P} \left(\delta_{c,p} \cos\left(2\pi p \frac{d(t)}{365} \right) + \delta_{s,p} \sin\left(2\pi p \frac{d(t)}{365} \right) \right) \tag{6.60}$$

$$\sigma_t^2 = \sum_{q=1}^{Q} \left(\gamma_{c,q} \cos\left(2\pi q \frac{d(t)}{365} \right) + \gamma_{s,q} \sin\left(2\pi q \frac{d(t)}{365} \right) \right) + \sum_{r=1}^{R} \alpha_r \varepsilon_{t-r}^2 \tag{6.61}$$

$$\varepsilon_t \overset{iid}{\sim} N(0,1)$$

跟以前一样，其中 $d(t)$ 是周期为 1，…，365 的重复跃阶函数（例如，一年的每一天假定为 1～365的一个值），并令 $L=25$，$P=3$，$Q=2$ 和 $R=1$.

模型（6.57）与模型（6.59）外加条件方差方程（6.61）是一样的，条件方差方程考虑了波动率动态特性的两种类型.首先，与方程（6.60）通过傅里叶级数在条件均值中近似条件季节性一样，通过在条件方差中近似季节性，它捕捉了季节性的波动率.[28]其次，方差方程捕捉了“条件方差移动中的自回归效应，它在时间序列背景下通常是自然产生的，对条件方差的冲击将对未来几期产生持续性的影响，与 Engle(1982) 创意性工作正好相同.”[29]使用 Engle(1982) 的非对称有效两步方法可以估计出模型.首先，方程（6.59）是采用普通最小二乘法估计得出的，在常数项、

趋势项、傅里叶项和滞后平均气温项上回归平均气温. 其次，在常数项、趋势项、傅里叶项和滞后平均气温项上回归方程（6.59）的方差残差方差方程（6.61）可以被估计出来. 拟合值$\hat{\sigma}_t^{-1}$逆的平方根被用来作为方程（6.59）的加权最小二乘再估计的权值.[30]

Campbell 和 Diebold(2002) 表明，条件异方差（heteroskedasticity）的估计模型减少了（但没有消灭）剩余超额峰度. 对于不同美国天气的日度平均气温，图 6.7 和图 6.8 分别表示相关曲线图和方差残差的相关曲线图.

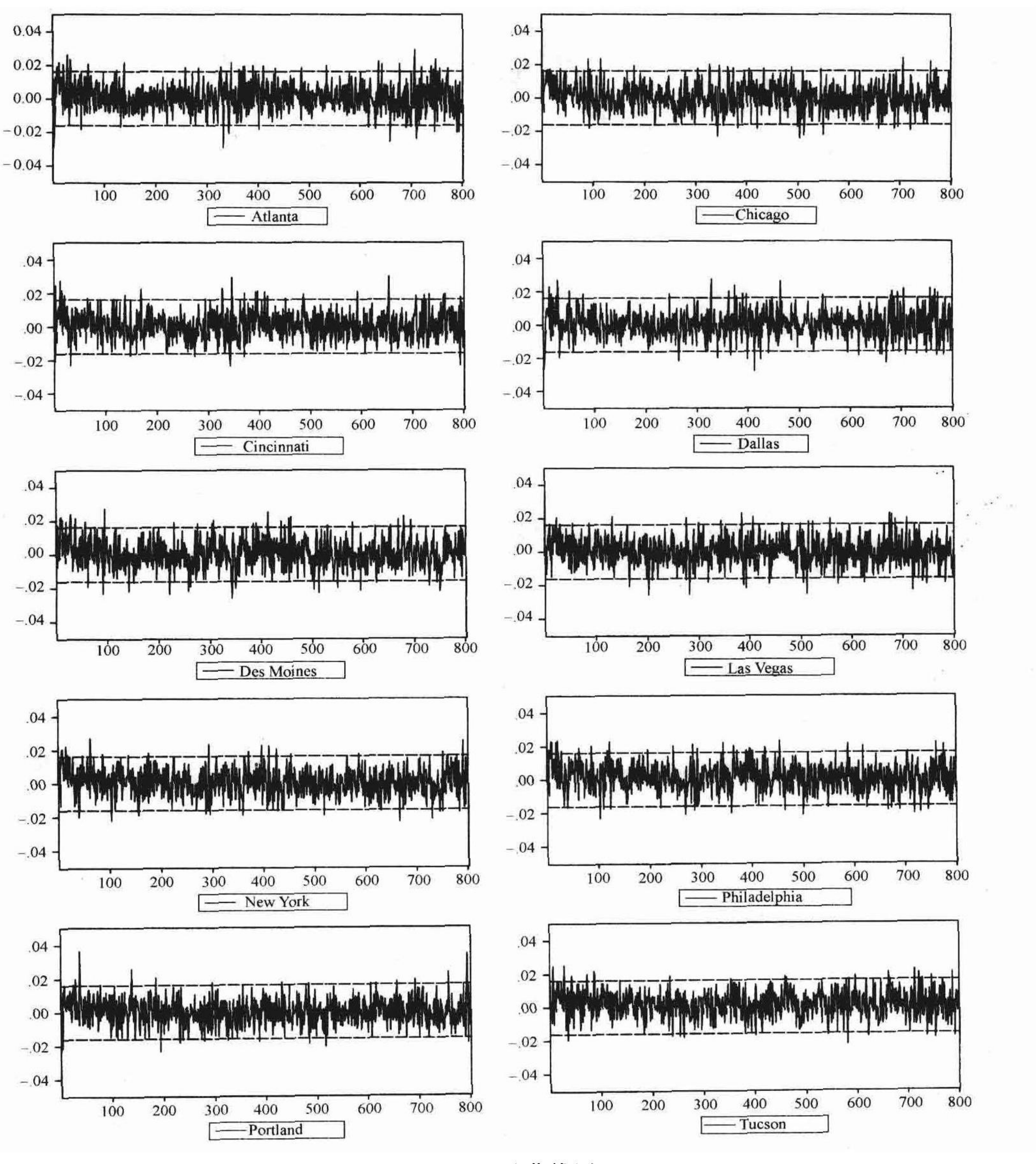


图 6.7 相关曲线图

资料来源：Campbell 和 Diebold(2002).

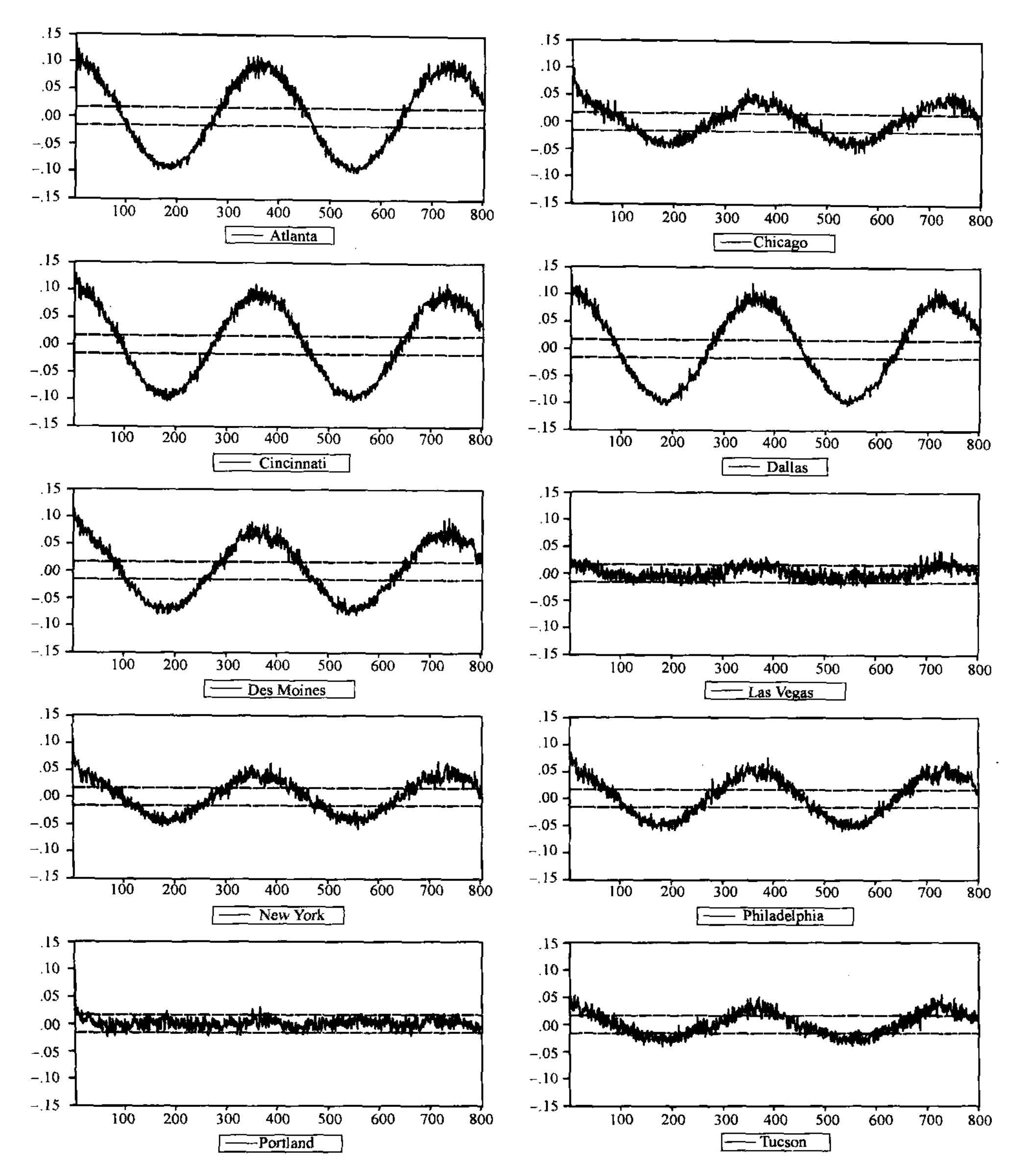


图 6.8 平方残差相关曲线图

资料来源：Campbell 和 Diebold(2002).

从我们的日度平均气温模型中，每幅图展示了平方残差的自相关性：

$$\left(T_t - Trend_t + Seasonal_t + T_t - Trend_t + Seasonal_t + \sum_{l=1}^{L} \rho_{t-1} T_{t-l}\right)^2$$

在白噪声原假设下，大约有 95%的置信区间.[31] 相关曲线图表明，没有证据表明在标准残差中存在序列相关.来自模型（6.59）的平方标准方差的相关曲线图是对模型（6.55）的极大提高，从白噪声行为表明这不存在显著的偏差，拟合模型（6.59）是足够的.[32]

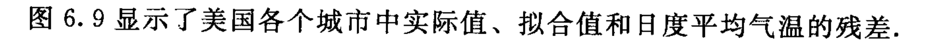


图 6.9 显示了美国各个城市中实际值、拟合值和日度平均气温的残差.

图 6.9　美国不同城市日度平均温度的实际值、拟合值和残差

资料来源：Campbell 和 Diebold(2002).

每幅图板展示了实际值、拟合值和来自如下不可观测项的残差：

$$T_t = Trend_t + Seasonal_t + \sum_{l=1}^{L} \rho_{t-1} T_{t-l} + \sigma_t \varepsilon_t$$

图 6.10 表明不同城市日度平均气温的估计条件标准差.

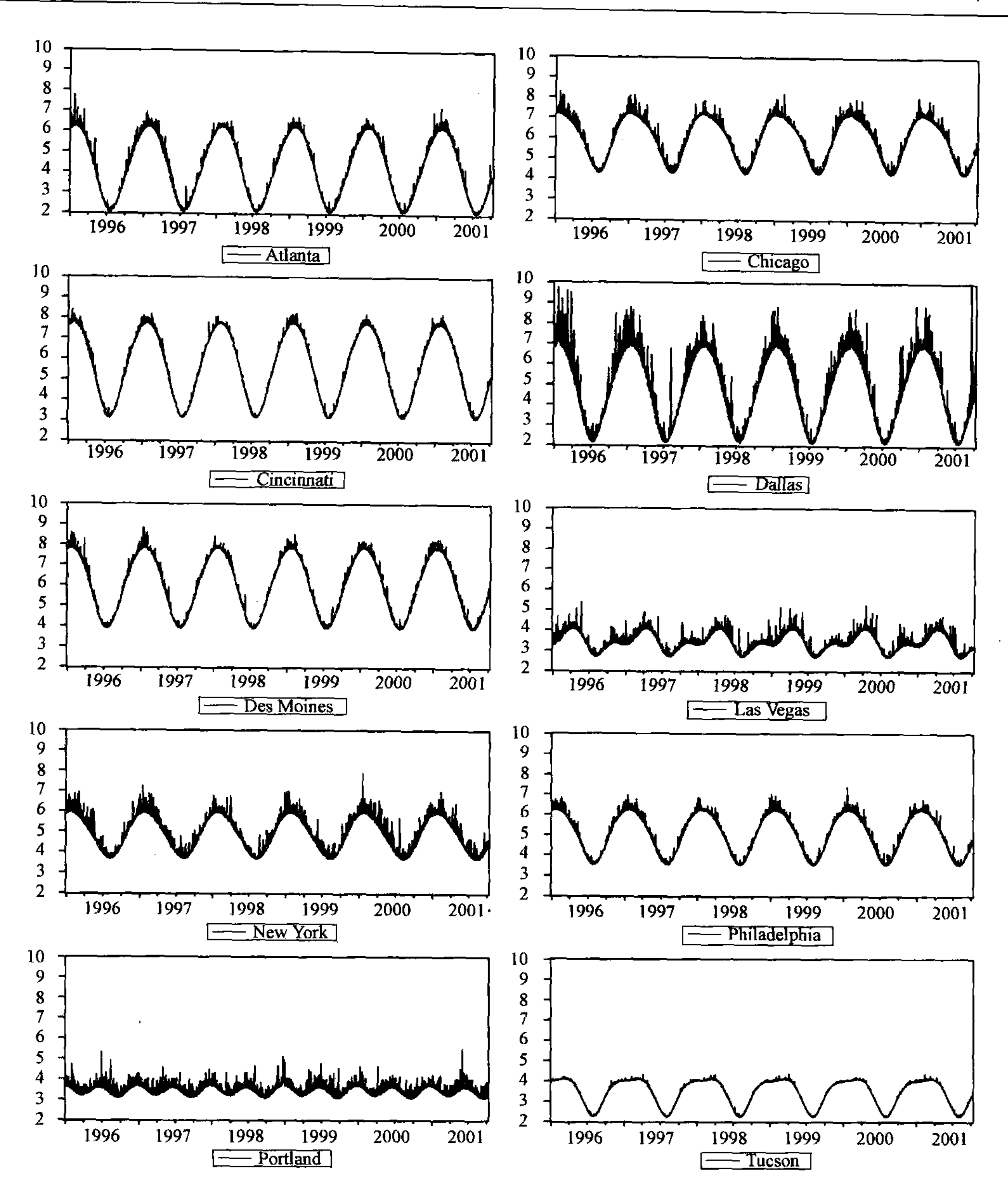


图 6.10 美国不同城市日度平均气温估计的条件标准差

资料来源：Campbell 和 Diebold (2002).

每幅图展示了从模型获得的日度平均气温的估计条件标准差的时间序列：

$$\hat{\sigma}_t = \sum_{q=1}^{2}\left(\hat{\gamma}_{c,q}\cos\left(2\pi q\frac{\mathrm{d}(t)}{365}\right) + \hat{\gamma}_{s,q}\sin\left(2\pi q\frac{\mathrm{d}(t)}{365}\right) + \hat{\alpha}\varepsilon_{t-1}^{2}\right.$$

图 6.11 表明日度平均气温估计的季节项（傅里叶级数与虚拟变量）.

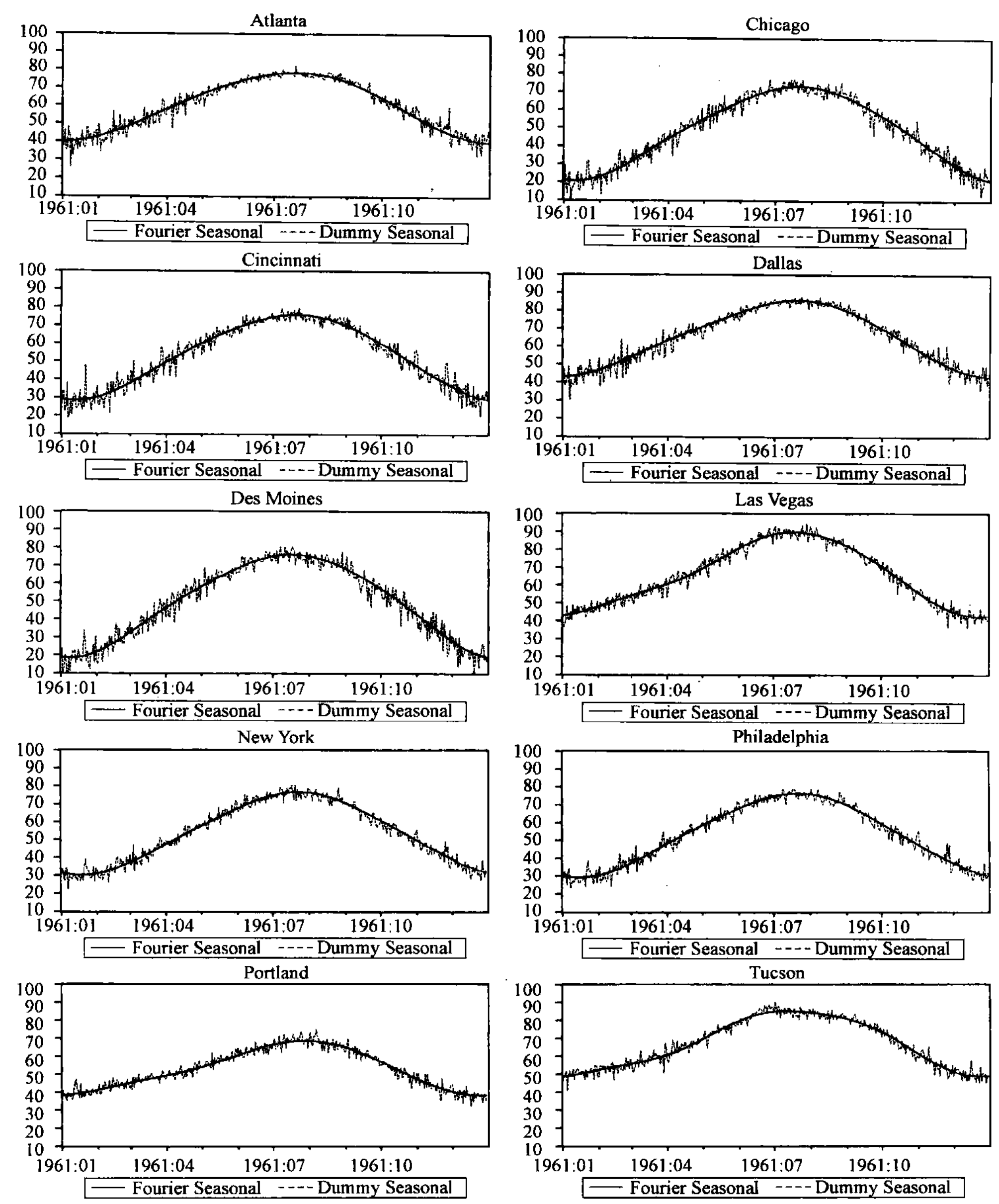


图 6.11　日度平均气温估计的季节项（傅里叶级数与日度虚拟变量）

资料来源：Campbell 和 Diebold (2002).

图片展示来自于傅里叶模型估计的平滑季节项，

$$Seasonal_t = \sum_{p=1}^{3}\left(\delta_{c,p}\cos\left(2\pi p\frac{d(t)}{365}\right)+\delta_{s,p}\sin\left(2\pi p\frac{d(t)}{365}\right)\right.$$

以及来自虚拟变量模型估计的粗糙季节项：

$$Seasonal_t = \sum_{t=1}^{365}\delta_i D_{it}$$

图 6.12 展示 1996—2001 年，美国各城市估计的日度平均气温的非条件分布的直方图.

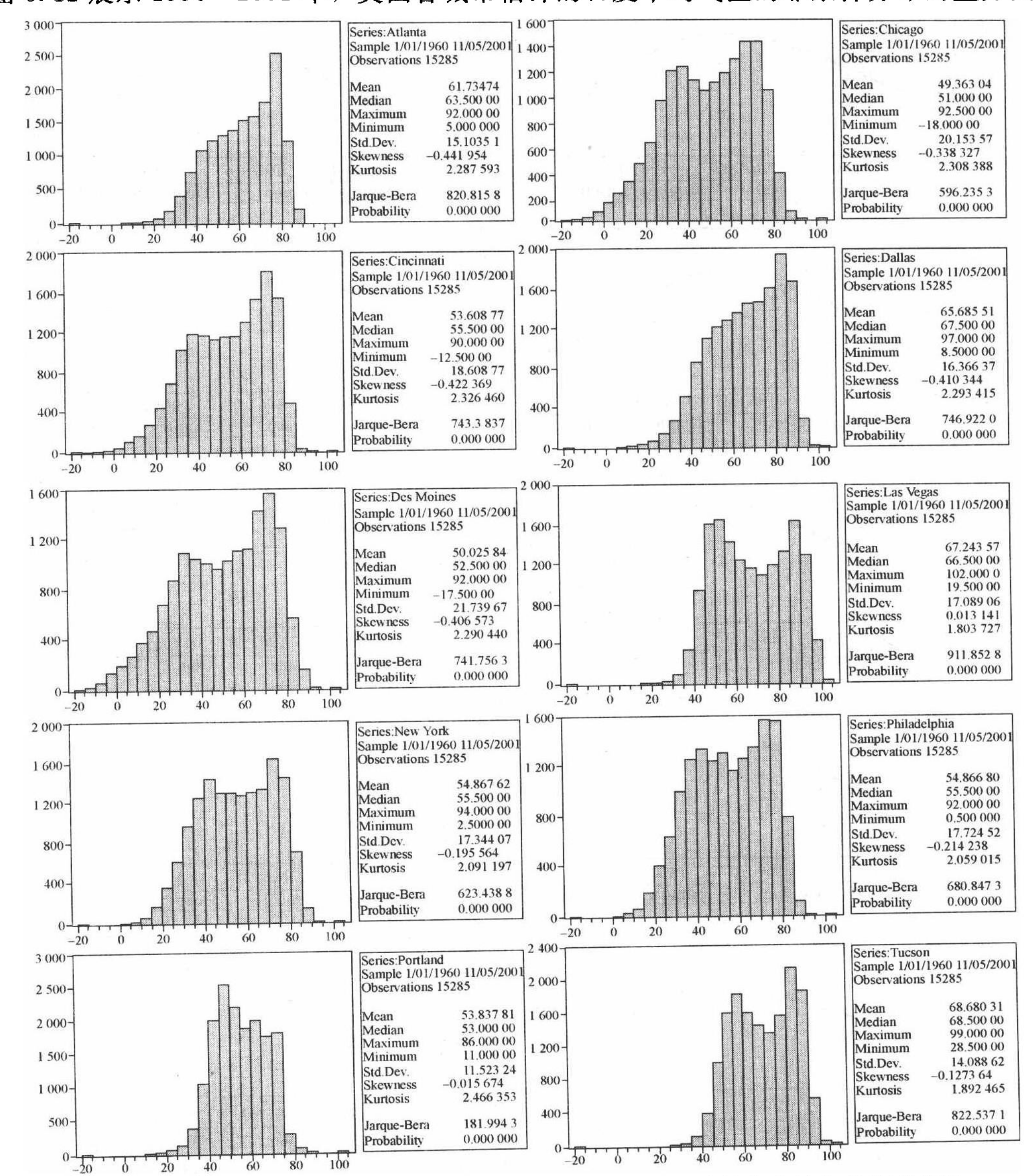


图 6.12 1996—2001 年，美国不同城市日度平均气温估计的非条件分布的直方图

资料来源：Campbell 和 Diebold(2002).

6.10 用 C++ 定价天气期权

天气期权（如 HDD 或 CDD）可以被视为亚洲期权，因为收益依赖于这一时期的平均气温. 我们可以用蒙特卡罗方法来给这些期权定价. 我们可以使用计算亚洲期权的代码来定价天气期权，如 HDD，要利用公式（6.1）中的平均公式，并取公式（6.2）中的个体收益和公式（6.46）

中相关的总计收益. 对这一时期（M 天）中的每一天，我们需要模拟 N 次（这一天内气温的抽样次数）气温，并取计算最高温度、最低温度和平均温度. 这一程序被模拟了 K 次，并且平均收益覆盖了所有路径，贴现产生期望的价值. 我们使用 Ornstein-Uhlenbeck 过程来模拟天气过程.

MCPricer_calcMCAsianPriceWeather.cpp

```
vector<double> MCPricer::calcMCAAsianPriceWeather(double price,
  double strike, double DDstrike, double vol, double rate,
   double div, double T, char type, long K, long M, long N)
{
        // initialize variables
        double A = 0.0;                  // arithmetic average
        double mu = 0.0;                 // drift
        int i, j, k;                     // counters
        double deviate;                  // normal deviate
        double stddev = 0.0;             // standard deviation
        double stderror = 0.0;           // standard error
        double W = 0.0;                  // weather option price
        double sum = 0.0;                // sum payoffs
        double sum1 = 0.0;
        double sum2 = 0.0;               // sum squared payoffs
        double payoff = 0.0;             // payoff
        double val = 0.0;                // option value
        double dt = (double) T/N;        // compute step size
        double a = 0.10;                 // mean reversion
        double Wbar = 65;                // long-run temperature
        double minW, maxW = 0;           // min and max temperature
        double tick_size = 100;          // tick size
        vector<double> value;            // store price, std dev., etc.
        vector<double> Wvec;             // store temperatures
    mrng.sgenrand(unsigned(time(0)));    // initializer RNG

        // number of simulations
        for (k = 0; k < K; k++)
        {
                payoff = 0;
                // for each day
                for (i = 0; i < M; i++)
                {
                W = Wbar;
                        Wvec.clear();
                        Wvec.empty();
                        // number of time steps (hours) in each day
                for (j = 0; j < N; j++)
                {
                            deviate = mrng.genrand();
                                mu = -a*(W - Wbar);
                            W   = W + mu*dt + vol*sqrt(dt)*deviate;
                        // Ornstein-Uhlenbeck process
                                Wvec.push_back(W);
                }
                        // sort temperatures
                        sort(Wvec.begin(),Wvec.end());
                        minW = Wvec[0];
                        maxW = Wvec[Wvec.size()-1];
                A = 0.5*(maxW + minW);
                if (type == 'C')  // cooling days
                    payoff += tick_size*max(A - strike, 0);
                else
                    payoff += tick_size*max(strike - A,0);
```

```
        }
        sum += payoff;
        sum2 += payoff*payoff;
        }
        sum = (double) sum/(K*M);  // average over all pays and days
        sum2 = sum*sum;
        val = exp(-rate*T)*(sum);
        value.push_back(val);

        stddev = sqrt((sum2 - sum*sum/M)*exp(-2*rate*T)/(K-1));
        value.push_back(stddev);

        stderror = stddev/sqrt(K);
        value.push_back(stderror);

        value.push_back(payoff);

        if (type == 'C')
                value.push_back(tick_size*(DDstrike - payoff));
        else
                value.push_back(tick_size*(payoff - DDstrike));

    return value;
}
```

主要方法是：

MCPricer_weather_main.cpp

```
#include "MCPricer.h"

void main()
{
    MCPricer mcp;
    long N = 5;
    long M = 100000;

    // weather options
    double W = 50;
    double strikeW = 65;
    double volW = 0.2;
    long numDays = 30;
    long K = 10000;
    long numSteps = 100;
    double HDDstrike = numDays*strikeW;
    double mat = (double) numDays/360;
    vector<double> val;
    cout.precision(8);

    val = mcp.calcMCAAsianPriceWeather(W,strikeW,HDDstrike,volW,rate,div,
       mat,'C',K,numDays,numSteps);
    std::cout << "Weather option price = " << val[0] << endl;
    std::cout << "Std deviation     = " << val[1] << endl;
    std::cout << "Std Error =          " << val[2] << endl;
    std::cout << "Actual HDD Price = "<< val[3] << endl;
    std::cout << "HDD Payoff at Maturity = " << val[4] << endl << endl;
}
```

HDD 的定价，其均值回复为 0.1，波动率为 0.2，长期均值为 65F，$K=10\,000$，$M=30$，$N=100$.

```
Weather option price = 14.462588
Std deviation = 0.14220213
Std Error = 0.0014220213
Actual HDD Price = 433.15165
HDD Payoff at Maturity = 151684.83
```

尾注

1. Platen and West(2004), 2.
2. Alaton P, Djehiche B, and Sillberger D(2000), 1.
3. Cao, Li, and Wei(2004), 1.
4. Benth F, Salyte-Benth J(2005), 9.
5. Id. 9.
6. Alaton P, Djehiche B, Sillberger D(2000), 9.
7. Id. 9.
8. Dornier and Queruel (2002).
9. Alaton P, Djehiche B, and Sillberger D(2000), 10.
10. Basawa and Prasaka Rao(1980), 212～213.
11. Brockwell and Davis(1990).
12. Bibby and Sorenson(1995).
13. Alaton P, Djehiche B, and Sillberger D(2000), 16.
14. Platen and west(2004), 23.
15. Alaton P, Djehiche B, and Sillberger D(2000), 17.
16. Zeng L(2000), 77.
17. 这些在第一列之后的 6 列中报告、最后 3 列表明在不同抽样长度下估计的期权价值.
18. Cao, Li, and wei(2004), 14.
19. Id. 14.
20. Id. 14.
21. Cao, Li, and wei(2004), 15.
22. Campbell S, and Diebold X(2000), 5.
23. Id. 5.
24. Campbell, S. and Diebold, F. (2000), 6.
25. Id. 6.
26. Id. 6.
27. Id. 6.
28. Campbell, S and Diebold F(2002), 9.
29. Engle(1982).
30. Campbell S and Diebold F(2002), 9.
31. Id.
32. Id. 10.

第7章 能源与电力衍生品

本章将详细探讨能源和电力衍生品的定价. 7.1 节讨论电力市场. 7.2 节回顾电力定价模型，包括单因素与双因素模型、跳跃扩散模型和随机波动模型. 还将讨论模型参数的估计. 7.3 节涉及电力摆动期权. 7.4 节探讨利用 Longstaff-Schwartz 最小二乘蒙特卡罗算法（LSM）定价美式和百慕大期权. 7.5 节在 Doerr(2003) 和 Meyer(2004) 工作的基础上扩展 LSM 的应用至摆动期权定价. 7.6 节把向上摆动、向下摆动和惩罚函数、摆动期权的通常特性与 LSM 定价算法相结合. 7.7 节是关于应用 Matlab 定价摆动期权. 7.8 节提供了源于 Doerr(2003) 的 LSM 模拟结果. 7.9 节探讨能源商品衍生品的定价，包括跨商品价差（Cross-Commodity spread）期权，以及裂解和点火价差期权. 7.10 节讨论跳跃扩散模型定价电力衍生品，而在 7.11 节讨论随机波动电力定价模型. 7.12 节基于 Xiong(2004) 的工作，探讨前两节中定价模型的参数估计，还将讨论最大似然（ML）、广义矩法（GMM）、条件特征函数（CCF)的 ML、光谱广义矩法（SGMM）等估计方法. 7.13 节讨论用 Matlab 实现 Xiong(2004) 的参数估计法. 7.14 节回顾常用能源商品定价模型. 7.15 节讨论天然气衍生品及市场概况. 7.16 节在 Xu(2004) 的工作基础上探讨定价模型. 7.17 节讨论应用 Matlab 定价天然气. 7.18 节探讨天然气和电力掉期.

7.1 电力市场

就像天气一样，电力具有不可储藏性以及有限的可运输性特征，这使得在不同时间和不同日期交割的电力被消费者认为是不同商品. 在不同时期（如季节性的），电力的尖峰需求和非尖峰需求对电价影响巨大，并且对由电力需求决定的电力市场也非常重要，例如，衍生出契约形式. 电价与消费者的电力需求及其决定因素（包括商业活动和当时天气状况）密切相关.

电的不可储藏和有限运输的性质影响了跨时空"携带"电力的能力. 与其他商品相比较，电的不可储藏和有限运输的性质成为解释电力现货和远期衍生品价格行为的关键. 换句话说，"在电力市场中，基于可储藏和可运输特性的跨时空套利即使未被完全消除，也被严格限制了."[1] 如果由套利导致的时空之间的联系被割裂，我们可以预期现货价格与当时和局部供求状况高度相关.[2]

因此，套利的有限性预计会深远地影响现货价格与远期价格之间的关系. 不可储藏性意味着套利的观点不能应用于定义定价模型，而这个定价模型中电力像天气一样成为衍生合约的标的资产. 结果，电力作为一种不可储藏商品，具有物理约束和暂时约束，因为这些定价模型不能捕捉这些限制，所以持有成本模型不再有效（参见 German 和 Roncoroni[2001]).

传输线路的容量和运输负荷对电力的传输施加限制，使得电力传输到一些区域难以实现或者经济上不划算. 电力供应同样取决于传输线路连接的可实现性，罕见、极端的事件，如发电厂故障——例如，2003 年 8 月发生在美国中西部和北部的事件导致严重的电力中断，这些事件能够阻碍电力供应. 这样的电力故障会引致明显的价格脉冲. 1998 年的夏天，由于未预期到一些发电厂和关键传输线路的拥堵，美国东部和中西部的电力现货价格从 50 美元/MWh 蹿升至 70 美元/MWh. 举一个例子，得克萨斯州（ERCOT）的历史尖峰电力现货价格以及加利福尼亚州和俄勒冈州交界处（COB）的历史尖峰电力现货价格如图 7.1 所示. 电力现货价格的跳跃行为主要归因于："电力的典型地区加总供应函数几乎总是在一定容量水平弯曲，超出那个

容量水平后，供应曲线上升斜率非常陡峭.”[3]

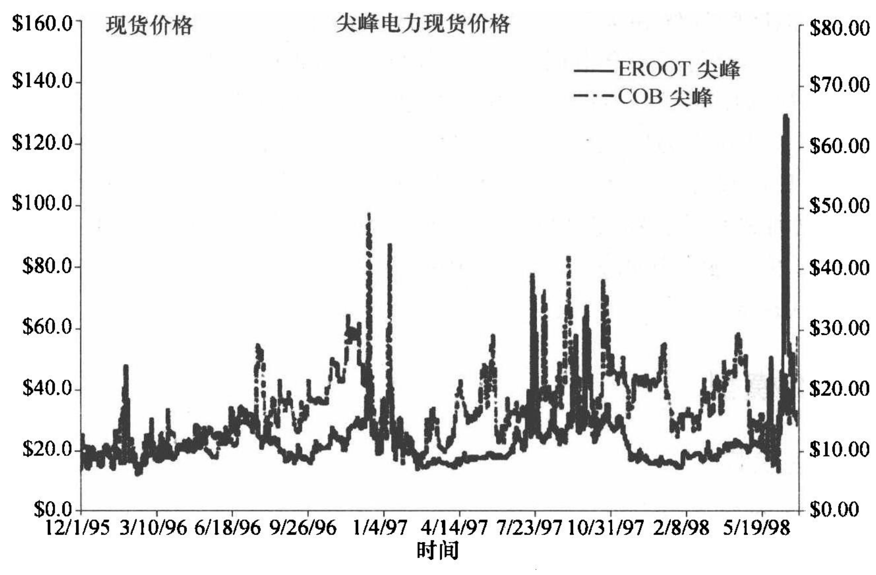


图 7.1 ERCOT 和 COB 尖峰电力现货价格

图 7.2 展示了 1999 年至 2002 年美国中西部（ECAR）、宾夕法尼亚州-马里兰州-新泽西州（PJM）区域、加利福尼亚州和俄勒冈州交界处以及输电至加利福尼亚州的一个主要中转站 Palo Verde 批发电力价格的比较.结果表明，在夏季的月份会出现大量的“尖峰”.

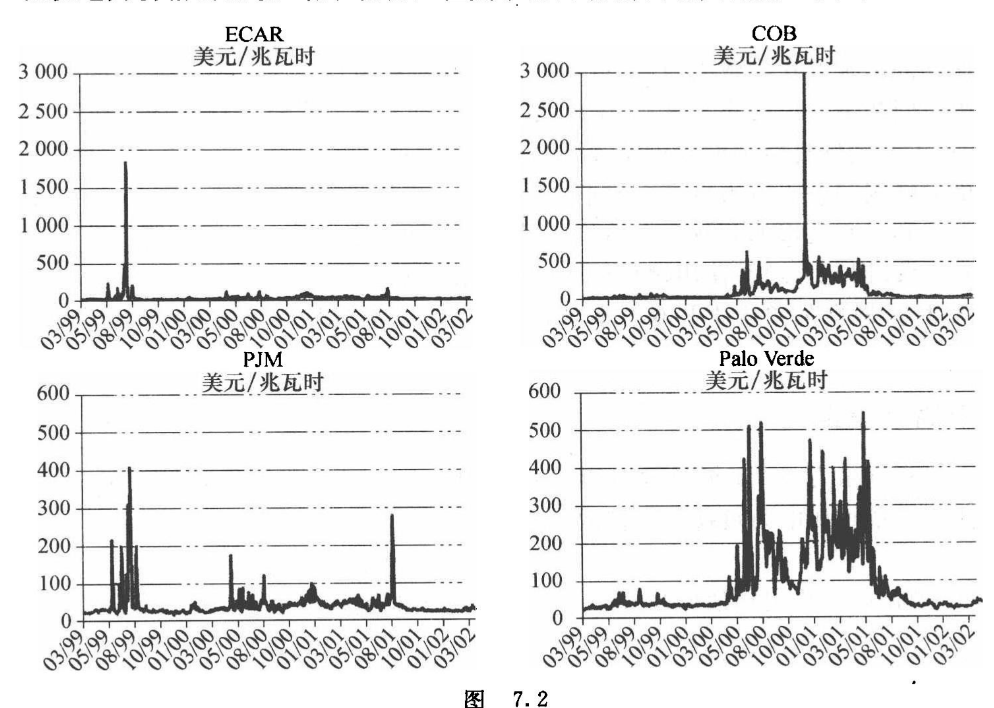


图 7.2

资料来源：商品期货交易委员会（参见美国能源情报署，EIAGIS-NG 地理信息系统）.

这些限制使得电力合约和价格“高度地区化；比如，与当时的供需决定因素高度相关（例如，当时发电厂、地区的气候和天气状况的特性以及产生的电力消耗）.”[4] 由于电力的不可储藏性和传输受限制，一般在不完全市场框架下完成衍生出来的定价方法.

撤销能源市场管制加剧了电力批发市场的竞争，电力衍生品合约也随之出现.这些合约既在场外交易（OTC）又在交易所交易，它们种类众多，可以满足电力市场参与者的需求.在美国，电力期货合约和期权合约近些年来出现在芝加哥商品交易所（CBOT）、纽约商品交易所（NYMEX）和明尼阿波利斯谷物交易所.[5] 基于能源和发电设备投资预期增加的需求，这些合约的需求得以增加.美国能源情报署（EIA）预测，为了满足美国未来 10 年的电力需求，需要约 1980 亿瓦新发电容量.[6]

7.2 电力定价模型

7.2.1 定价过程建模

能源价格主要由供需关系驱动.基于电力的一些特性，如它生产出来以后在时间和地点方面的同质特性、它的不可储藏性以及考虑到套利方式制约导致的电力不完全市场，电力现货价格具有短期波动性特征.另外，供需特性决定了电力价格具有如下特征：均值回归、周期波动和随机价格尖峰.

在均值回归的情形下，波动率随着时间范围的增加而降低.存在一个长期均衡（“公平价格”），它比现货价格波动小得多.均值回归的速度取决于供应能多迅速地反映突然的需求变化（参见 Pilipovic(1998)）.[7] 在不同的时间标度（日度、周度、季度）都会出现周期波动，而周期波动是周期性的需求变化导致的.价格过程的周期波动性被认为是具有决定性的，因而能够轻易地从随机时间相关性中分离出来.[8] 除大的短期波动之外，价格尖峰偶尔发生，但仅仅持续很短的时间.正的尖峰是由供电或传输过程的损耗（参见 Eydeland 和 German[1999]）导致的，而负的价格尖峰则归咎于电力低需求时期很难减少发电量.

如果考虑到这些尖峰的观测，电力价格过程并不适合于用简单几何布朗运动建模，因为它们无法刻画尖峰.因此，通常使用跳跃扩散过程建模.例如，可以在对数正态模型中加入一个离散跳跃扩散部分.跳跃的单因素和双因素对数正态均值回复过程已经被提出.下面我们将检验单因素模型.

7.2.2 单因素模型

记电力现货价格为 S_t.随机过程可以表示为两部分之和——关于时间的确定性函数 $f(t)$ 和一个单状态变量（如电力负荷）的扩散随机过程 X_t，即

$$S_t = f(t) + X_t \tag{7.1}$$

假定 X_t 遵循一个稳定均值回复 Ornstein-Uhlenbeck 过程：

$$dX_t = -\kappa X_t dt + \sigma dW_t \tag{7.2}$$

其中，$\kappa>0$ 表示均值回复的速度，$X(0)=x_0$，以及 dW_t 表示标准布朗运动的一个增量.因为 $X_t=S_t-f(t)$，并假定函数 $f(t)$ 满足适当的正则条件，可将公式（7.1）和公式（7.2）重写为：

$$d(S_t - f(t)) = \kappa(f(t) - S_t)dt + \sigma dW \tag{7.3}$$

这表明当 S_t 偏离确定性函数 $f(t)$ 时，它被以一个与其偏差成比例的速度拉回.在这个模型中，不确定性仅仅来源于随机行为 X_t，正如公式（7.2）所描述的那样.

采用 Lucia 和 Schwartz(2001) 的方法，S_t 过程可以被表示为随机微分方程的解（假定函数 $f(t)$ 满足适当正则条件，例如 $\int_{-\infty}^{\infty} f(t)^2 dt < \infty$，假定函数是有界的)，如下所示：

$$dS_t = \kappa(\alpha(t) - S_t)dt + \sigma dW \tag{7.4}$$

其中，$\alpha(t)$ 是 t 的确定性函数，被定义为：

$$\alpha(t) = \frac{1}{\kappa}\frac{df(t)}{dt} + f(t) \tag{7.5}$$

它可以被看成是扩展 Vasicek 模型的一种特殊情形.

假定的简单单因素模型比较容易解析.与公式（7.1）联立，公式（7.2）的一个显性解为：

$$S_t = f(t) + X_0 e^{-\kappa t} + \sigma\int_0^t e^{\kappa(s-t)} dW(s) \tag{7.6}$$

我们发现 E_t 的条件分布是正态的，它的条件均值和方差为（令 $X_0 = E_0 - f(0)$）：

$$E[S_t] = E[S_t | X_t] = f(t) + (S_0 - f(0))e^{-\kappa t} \tag{7.7}$$

$$\mathrm{Var}[S_t] = \mathrm{Var}[S_t \mid X_0] = \frac{\sigma^2}{2\kappa}(1 - e^{-2\kappa t}), \quad \kappa > 0$$

其中，$E[\cdot]$ 是期望运算符.

从长期来看，当给定 S_0 初值时，价格过程 S_t 趋向于均值 $f(t)$. κ(假定 $\kappa>0$) 值越高，收敛越快.方差反过来又随着时间范围减少，并且当时间范围趋向于无穷时，方差有一个极值为 $\sigma^2/2\kappa$.

为了对电力衍生品定价，我们需要状态变量 X_t 基于鞅测度的风险中性过程，而不是公式（7.2）中物理测度的实际过程.对于 X_t 的非交易特性，两个衍生资产的标准套利观点使我们能够获得 X_t 的风险中性过程.运用 Girsonov 变换后，通过令 $dW^* = dW - \lambda dt$，我们改变了漂移项，所以有：

$$dX_t = \kappa(\alpha^* - X_t)dt + \sigma dW^* \tag{7.8}$$

其中，

$$\alpha^* = -\frac{\lambda\sigma}{\kappa} \tag{7.9}$$

dW^* 是 W_t^* 的一个增量，是风险中性概率测度下的一个标准布朗运动，λ 表示与状态变量 X_t 有关的风险的市场价格.当假定 λ 不变时，定价公式是 t 和状态变量 X_t 的函数.

公式（7.8）中 SDE 的显性解为：

$$S_t = f(t) + X_0 e^{-\kappa t} + \alpha^*(1 - e^{-\kappa t}) + \sigma\int_0^t e^{\kappa(s-t)} dW^*(s) \tag{7.10}$$

α^* 如式（7.9）所定义的那样.我们发现 S_t 在风险中性测度下是条件正态分布，条件均值定义如下：

$$E^*[S_t] = f(t) + X_0 e^{-\kappa t} + \alpha^*(1 - e^{-\kappa t}) \tag{7.11}$$

我们知道任何衍生资产的价值是其回报的预期值，并在风险中性测度下以不变的无风险利

率贴现到估值日. 到期日为 T 的电力远期合约在零时刻的现货价格必然为：

$$V_0(X_t,T) = e^{-rT}E_0^*[S_T - F_0(S_0,T)] \tag{7.12}$$

其中 $F_0(X_0, T)$ 是合约到期日为 T、在时刻零的远期价格，r 是无风险连续复利利率. 如果从一开始就进入，远期合约的价值必然为零，通过运用公式（7.11）和（7.1）并令 $t=0$，最终得出电力远期价格的封闭解[9]：

$$F_0(S_0,T) = E_0^*[S_T] = f(T) + (S_0 - f(0))e^{-\kappa T} + \alpha^*(1 - e^{-\kappa T}) \tag{7.13}$$

其中，$\alpha^* = -\lambda\sigma/\kappa$.

假设公式（7.1）中的模型被修改成使用现货价格的自然对数形式，而不是现货价格本身，如下所示：

$$\ln S_t = f(t) + Y_t \tag{7.14}$$

可得：

$$S_t = f(t)e^{Y_t}$$

其中，$f(t)$ 是已知关于时间的确定性函数，Y_t 是随机过程，它的动态性给定如下：

$$dY_t = -\kappa Y_t dt + \sigma dW \tag{7.15}$$

其中，$\kappa>0$，$Y(0)=y_0$. 对数价格遵循均值为 0 的均值回复过程，这意味着 $f(t)$ 在一定条件下服从如下的价格过程：

$$dS_t = \kappa(b(t) - \ln S_t)S_t dt + \sigma S_t dW(t) \tag{7.16}$$

其中，

$$b(t) = \frac{1}{\kappa}\left(\frac{\sigma^2}{2} + \frac{d\log f(t)}{dt}\right) + \log f(t)$$

由公式（7.16）可得，$\ln S_t$ 服从条件正态分布，其条件均值和条件方差分别为：

$$\begin{aligned} E_0[S_t] &= \exp\left(E_0[\ln S_t] + \frac{1}{2}\text{Var}_0[\ln S_t]\right) \\ &= \exp\left((f(t) + (\ln S_0 - f(0))e^{-\kappa t} + \frac{\sigma^2}{4\kappa}(1 - e^{-2\kappa t})\right) \end{aligned}$$

和

$$\begin{aligned} \text{Var}_0(S_t) &= \exp(2E_0[\ln S_t] + \text{Var}_0[\ln S_t])(\exp(\text{Var}_0[\ln S_t]) - 1) \\ &= E_0[S_t]^2\left[\exp\left(\frac{\sigma^2}{2\kappa}(1 - e^{-2\kappa t})\right) - 1\right] \end{aligned}$$

运用 Girsonov 理论把对数价格过程转换成风险中性测度以后，与公式（7.8）类似，对数电力价格的远期合约为[10]：

$$\begin{aligned} F_0(S_0,T) &= E_0^*[S_T] \\ &= \exp\left[f(T) + (\ln S_0 - f(0))e^{-\kappa T} + \alpha^*(1 - e^{-\kappa T}) + \frac{\sigma^2}{4\kappa}(1 - e^{-2\kappa T})\right] \end{aligned} \tag{7.17}$$

其中，$\alpha^* = -\lambda\sigma/\kappa$. 注意，现货价格（对数价格）行为的确定性构成直接体现在远期（期货）合约价格中（可参见公式（7.13）和公式（7.17)），该式“成为远期（期货）价格曲线形状的一个重要确定性因素”.[11]无论是单因素模型还是对数因素模型，所有远期（期货）价格都完全相关.[12]

7.2.3 估计确定性部分

为了完善公式（7.1）和公式（7.14）中的模型，有必要阐明确定性时间函数 $f(t)$. 这个函数试图捕捉电力价格行为的任何相关可预测部分，这种电力价格行为随时间变动源自真正的正则性. 尽管有多种选择，但是被选择的函数应该有一个确定性的一般趋势捕捉季节和周期性行为. 例如，正如 Pilipovic(1998) 所建议的那样，正弦函数和余弦函数都可以用来反映价格时间序列的一般季节模式.

Lucia 和 Schwartz(2001) 推荐了如下函数：

$$f(t)=\alpha+\beta D_t+\gamma\cos\left((t+\tau)\frac{2\pi}{365}\right) \tag{7.18}$$

其中，

$$D_t=\begin{cases}1, & \text{如果 } t \text{ 是周末或节假日}\\ 0, & \text{其他}\end{cases}$$

cos 是用弧度表示的余弦函数，α，β，γ 和 τ 全是不变参数. 这里，系数 β 试图捕获周末和节假日变量的变化，因为在周末和节假日用电量会明显增加. 余弦函数是为了反映季节模式，而通过演化到年度相关变量，余弦函数同样可用来反映年度周期.

7.2.4 估计单因素模型的随机过程

为了通过现货价格数据估计单因素模型的随机过程，将公式（7.1）离散化：

$$X_t=(1-\kappa)X_{t-1}+\xi_t \tag{7.19}$$

对于 $t=0, 1, 2, \cdots, N$，其中更新 ξ_t 是均值为 0、方差为 σ^2 的独立同分布正态随机变量. 同样的离散化方式可用来处理公式（7.15）中的过程 Y_t.

离散化以后，可以通过单因素价格模型和对数价格模型估计参数：

单因素模型

$$\begin{aligned}S_t&=\alpha+\beta D_t+\gamma\cos\left((t+\tau)\frac{2\pi}{365}\right)+X_t\\ X_t&=\phi X_{t-1}+u_t\end{aligned} \tag{7.20}$$

单因素对数价格模型

$$\begin{aligned}\ln S_t&=\alpha+\beta D_t+\gamma\cos\left((t+\tau)\frac{2\pi}{365}\right)+Y_t\\ Y_t&=\phi Y_{t-1}+u_t\end{aligned} \tag{7.21}$$

D_t 是公式（7.18）所定义的虚拟变量，且 $\phi=1-\kappa$. 对于上述两个模型来说，参数估计都是同时使用非线性最小二乘法. 规范一下，可以把这些模型写成统一的形式：

$$\begin{aligned}y_t&=f(\boldsymbol{\Phi},x_t)+\xi_t\\ \xi_t&=\phi\xi_{t-1}+u_t\end{aligned} \tag{7.22}$$

第一个方程描述了因变量 y_t（即价格变量或对数价格变量）作为参数向量 $\boldsymbol{\Phi}$ 和解释变量向量 $\boldsymbol{x}_t$ 的一个函数. 第二个方程是第一个方程中扰动项 ξ_t 的一阶自回归方程. 把 ξ_t 替换到第二个方程中，重新排列后可得：

$$y_t = \phi y_{t-1} + f(\boldsymbol{\Phi}, x_t) - \phi f(\boldsymbol{\Phi}, x_{t-1}) + u_t \tag{7.23}$$

我们使用非线性最小二乘法同时估计参数 ϕ 和 $\boldsymbol{\Phi}$.[13]最终，将$\hat{\kappa}=1-\hat{\phi}$作为均值回复参数$\kappa$ 的估计，回归标准误差作为 σ 的估计. 基于 Nordic 电力交易所[14]从 1993 年 1 月 1 日至 1999 年 12 月 31 日的日度电力价格，Lucia 和 Schwartz(2005) 使用这个程序估计了这些模型的参数（双因素模型也一样），如表 7.1 所示.

表 7.1

参数	单因素价格模型				对数价格模型			
	模型 1		模型 2		模型 3		模型 4	
	估计	t-统计	估计	t-统计	估计	t-统计	估计	t-统计
α	153.051	8.146	145.732	8.670	4.938	38.711	4.867	46.192
β	−9.514	−28.085	−9.542	−28.277	−0.090	−28.339	−0.090	−28.523
γ			29.735	2.336			0.306	2.986
τ			6.691	0.269			0.836	0.043
β_2	−2.527	−0.754			−0.027	−0.878		
β_3	−4.511	−0.998			−0.041	−0.977		
β_4	−3.484	−0.664			−0.041	−0.849		
β_5	−13.248	−2.317			−0.185	−3.480		
β_6	−12.656	−2.114			−0.097	−1.744		
β_7	−7.038	−1.157			−0.062	−1.093		
β_8	−8.109	−1.347			−0.101	−1.807		
β_9	−10.061	−1.740			−0.094	−1.749		
β_{10}	−9.597	−1.795			−0.067	−1.352		
β_{11}	−7.304	−1.566			−0.052	−1.190		
β_{12}	−6.019	−1.674			−0.057	−1.703		
ϕ	0.990	355.4	0.989	340.4	0.986	299.0	0.984	277.5
κ	0.010		0.011		0.014		0.016	
回归标准误差	9.001		9.222		0.086		0.086	
调整的 R^2	0.981		0.981		0.974		0.973	
对数似然	−9294.2		−9299.0		2652.9		2640.2	
误差：								
M. A. E.	5.847		5.855		0.053		0.053	
M. A. P. E.	4.980		5.000		1.176		1.179	

资料来源：Lucia J 和 Schwartz E(2001)，32.

模型 1 和模型 3 仅仅包含了虚拟变量并各自剔除了价格和对数价格的余弦周期部分.[15]在 4 个模型中，独立系数 α 显著不等于 0. 系数 β、ϕ 和 σ 实际上在模型 1 和 2，模型 3 和 4 中难以被分辨. $\phi=1$ 的零假设（null hypothesis）在每个模型中被常用的 t 检验拒绝. 这意味着回复系数 κ 的估计值虽然很小，但在各种情形下都是显著的. 如预期的那样，系数 β 对应的虚拟变量 D_t 是负的，在 4 个模型中都不等于零，但并不是所有月度虚拟变量都是显著的.[16]

图 7.3 和图 7.4 画出了对应于拟合模型估计的实际日度系统价格以及等式（7.23）相应的



残差，也可以称之为一期向前预测误差（one period-ahead prediction errors）.

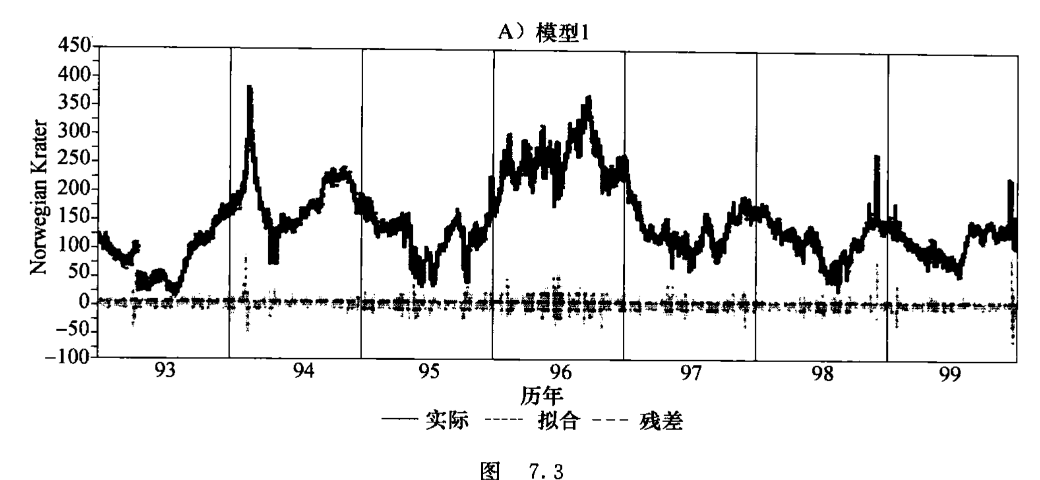


图 7.3

资料来源：Lucia J 和 Schwartz E(2002). 经 Review of Dervatives Research 允许后重印.

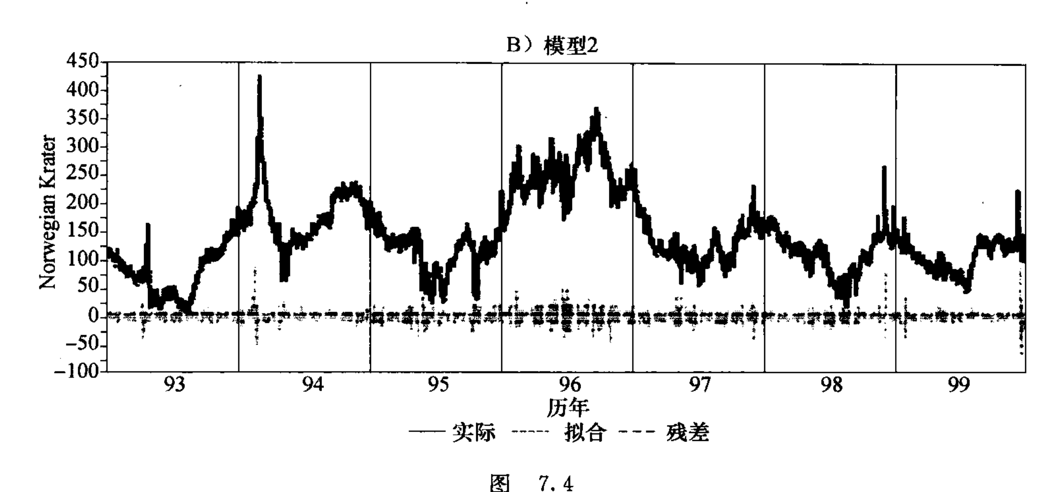


图 7.4

资料来源：Lucia J 和 Schwartz E(2002). 经 Review of Dervatives Research 允许后重印.

7.2.5 双因素模型

单因素模型可以拓展至双因素模型. 根据 Hillard 和 Reis(1998)，可以用双因素模型对电力价格建模：

$$
\begin{aligned}
dS_t &= (r-\delta_t)S_t dt + \sigma_S S_t dW_t^S \\
d\delta_t &= \alpha(\kappa_\delta - \delta_t)dt + \sigma_\delta dW_t^\delta
\end{aligned}
\tag{7.24}
$$

其中，r 表示利率，σ_S 是几何布朗运动 S 的方差，δ_t 是遵循均值回复（mean-reverting）过程的随机便利收益. 用 σ_δ 表示随机便利收益的方差，α_δ 是调整的速度，κ_δ 是长期均值收益. W_t^S 和 W_t^δ 两个维纳过程的相关系数为 ρ. 正如 Doerr(2003) 指出的，"便利收益这个概念仅仅被用在储藏商品上，这个模型更适合于用到天然气或者石油价格上，而不是电力价格."[17] 然而，正如下一章所讨论的那样，

它与摆动期权相关.进一步来说，净便利收益能被“解释为合并供需和电力市场的其他特殊性为一个变量的理论架构.”[18]这些效应“通常是随机的，这蕴含着便利收益随机行为的一个模型.作为模型的一个关键假设，电力因此被模型化为随机（正或负）分红收益率 δ_t 的资产，这个资产本身又服从均值回复 Ornstein-Uhlenbeck 过程.”[19]我们假定便利收益风险的市场价格是零.

7.3 摆动期权

为了对冲商品价格突变产生的市场风险，消费者可以对商品价格采用远期或期权.然而，对部分市场参与者来说，因为他们并不知道对商品的精确远期需求，仅运用远期和期权降低风险是不够的.特别地，这对于不可储藏商品（如能源与电力）是一个严重的问题，对于储藏费用昂贵的商品也是一个严重的问题.由于这个问题，开发了所谓的摆动期权（swing option），以使持有者对未来购买数量能够保持一定灵活性.[20]因为能源不可储存或者储存费用昂贵，并且会出现剧烈价格波动，所以摆动合约主要运用于能源市场，尤其是电力市场，但是摆动合约也出现在煤炭和天然气市场.[21]

我们将重点介绍电力的摆动期权.然而，摆动期权的主要特性，即多重提前执行特性，对于所有标的商品是一样的.典型的摆动期权包括所谓的基本负荷协议（参见 Jaillet、Ronn 和 Tompaidis[2001]).基本负荷协议是一系列不同到期日 $t_j(j=1, \cdots, N)$ 的远期合约.每一个远期合约 f_j 以电力（一般是任何商品）的一定数量 q_j 作为基准.在每一个到期日，持有者有权选择购买超额数量或减少基本负荷量.[22]这意味着摆动期权的持有者以预先约定的价格购买一定数量的电力能够在一定范围内（$q_j+\Delta_j$）摆动.如果 Δ_j 为正（负），那么持有者在机会时间 t_j 执行期权被称为向上摆动（向下摆动).因此，向上摆动是一种买入，而向下摆动是卖出.对于典型合约来说，有着更多的限制：对于一些边界 $N>0$，向上摆动的总量 U 和向下摆动总量 D 是有限制的，例如，$U\leqslant N$，$D\leqslant N$ 或 $U+D\leqslant N$.

如果在合约期限中所有买入成交量超出了预先规定的边界，那么摆动合约可能要承受一定的惩罚.一方面，这些额外的限制降低了摆动期权的价格.另一方面，它导致了期权不一样的执行策略.不一样的执行策略存在的原因在于，执行单一摆动权利的决定不仅仅取决于当时的电力价格.因为摆动权利的数量是受限制的，所以执行一个摆动权利会减少在未来可执行摆动权利的数目.另外，执行（不执行）有可能招致惩罚.因此，执行一个摆动权利后，它的报酬必须超过剩余期权的价值.故可得出结论，最优执行策略不仅取决于电力价格，而且取决于历史价格和未来价格的分布.

摆动期权与远期合约的组合称为摆动合约.摆动合约允许对合约的交易量添加灵活性.摆动合约的一个典型例子就是供应合约.这里，接收者有权利接受不超过一定最大负荷的任意电力交易量.惩罚的详情可参见“执行还是被惩罚”条款，这些条款规定无论接受了多少交易量，最小电力数目在任何情况下都必须被接受.这样的合约能被分解为摆动期权和远期合约.电力的供应者卖出隐含摆动期权.

我们直接采用 Doerr(2003) 和 Meyer(2004) 的合约，接下来讨论摆动期权的定价.

7.4 美式期权和百慕大期权的 Longstaff-Schwartz 算法

Longstaff-Schwartz (2001) (类似的方法也可参见 Clement 和 Protter (2002)) 详细阐述了

Longstaff-Schwartz 算法的基本思想. Longstaff-Schwartz 算法是在函数的有限集合上使用最小方差回归作为条件预期估计的替代. 第一步，时间轴必须被离散化，即如果在时间区间 $[0, T]$ 中，美式期权是存续的（*alive*），则期权仅在离散时间 $0<t_1<t_2<\cdots<t_J=T$ 可提前执行. 因此，美式期权近似于百慕大期权. 对于特定的执行日期 t_k，如果立即执行的价值超过存续价值（continuation value），即（剩余）期权的价值在 t_k 时刻未被执行的价值，则期权被提前执行. 这个存续价值可以被表示为与风险中性定价测度 Q 相关的期权报酬的条件期望. 这个期望是信息集$\Im_{t_k}$ 的条件期望，它可在时间 t_k 获得. 用 $F(\omega, t_k)$ 表示特定样本路径 ω 的存续价值，可以写成

$$F(\omega,t_k) = E^Q\left[\sum_{j=k+1}^{K} D(t_k,t_j)C(\omega,t_j,t_k,T)\,|\,\Im_{t_k}\right] \tag{7.25}$$

其中，$D(t_k, t_j)$ 是从时间 t_k 到 t_j 的贴现因子，$C(\omega, t_j, t_k, T)$ 表示期权产生的现金流的路径，条件是期权未在 t_k 或在 t_k 时刻以前被执行，并且对于 t_k 至 T 时刻之间的所有剩余机会 t_j，持有人遵循最优执行策略. 值得注意的是，对于每一条路径 ω，至少存在一个执行日 j，使得 $C(\omega, t_j, t_k, T)>0$，因为百慕大期权仅有一个执行权力. 在 t_{J-1} 执行的决定通过比较存续价值 $F(\omega, t_{J-1})$ 和当时的报酬 $P(S_{J-1})$ 做出，其中 S_{J-1} 是在 t_{J-1} 时刻的标的价值. 当 $P(S_{J-1})$ 已知时，存续价值必须被估计出来.

为了找出一个估计值，存续价值可以用一组基函数 B_j 表示：

$$F(\omega,t_{J-1}) = \sum_{i=0}^{\infty} a_i B_i(S_{J-1})$$

可近似表示为：

$$\hat{F}(\omega,t_{J-1}) = \sum_{i=0}^{M} a_i B_i(S_{J-1}) \tag{7.26}$$

系数 a_i 是用来回归 $C(\omega, t_J, t_{J-1}, T)$ 的贴现值到基函数上. 这些现金流发生在 t_J 时刻. 回归在具有存续价值的所有路径上均已完成，例如，期权在 t_J 时刻是价内的. $\hat{F}(\omega, t_{J-1})$ 是存续价值的无偏估计.

通过比较估计值$\hat{F}(\omega, t_{J-1})$ 与每条路径当时执行报酬 $P(S_{J-1})$ 来确定执行决定. 两者之间值较高者作为在时刻 t_{J-1} 的新现金流 $C(\omega, t_{J-2}, t_{J-1}, T)$，时间上向后迭代.

最终，期权的终值是通过取每条路径 ω 现金流的平均值来计算：

$$V_{LSM}^{N} = \frac{1}{N}\sum_{i=1}^{N} C(\omega_i) \tag{7.27}$$

这里，$C(\omega_i)$ 表示路径 ω_i 的贴现现金流.

LSM 算法

假设所有利率都是零，贴现就可以被省略了. 在开始真正的算法之前，形成标的现货价格的路径必须被作为样本. 对于 N 条路径和 J 个执行机会（时间步长），这会生成一个 $N\times J$ 矩阵 S，其中 $S_{i,j}$ 是第 i 条路径在时刻 t_j 的现货价格. 接下来，一组回归基函数 $(B_j)_{j=0}^{M}$ 必须从大概率中挑选出来，包括 Hermite、Legendre、Chebyshev、Gegenbauer 或 Jacobi 多项式.[23] 然而，Longstaff 和 Schwartz 强调 “数值检验表明傅里叶级数或三角级数甚至是状态变量的简单势也同样能给出精确的结果. 阶数 $M=2$ 的基函数（即二次多项式）在 LSM 算法中表现良好，并被 Doerr 使用.

$$B_0 = 1$$
$$B_1 = X$$
$$B_2 = X^2$$

真正算法的初始步是决定在最后一个时间步长 t_J 的现金流向量 C^J. 这些现金流比较容易获取，因为存续价值是零，即

$$C_i^J = P(S_{i,J}) \tag{7.28}$$

其中，P 是报酬函数. 接下来，我们关注单纯看涨期权：

$$P(S_{i,j}) = \max(S_{i,j} - X_j, 0) \tag{7.29}$$

其中，执行价格 X_j 随着时间步长而变化.

其次，考虑在时间步长 t_{J-1} 的现货价格，选择那些 $P(S_{i,J-1}) > 0$ 的值. 这会生成 $L_{J-1} \times 1$ 向量 $\hat{S}^{J-1}$，其中 L_{J-1} 是在时间步长 t_{J-1} 价内路径的数量. 通过最小化如下表达式，C^J 的最小方差回归到基函数 B_J 上可以实现：

$$\| B^{J-1} a^{J-1} - C^J \| \tag{7.30}$$

其中，a^{J-1} 是在时间步长 t_{J-1} 时回归系数的 $(M+1) \times 1$ 向量，矩阵 B^{J-1} 为：

$$B^{J-1} = \begin{pmatrix} B_0(\hat{S}_{1,J-1}) & \cdots & B_M(\hat{S}_{1,J-1}) \\ \vdots & \ddots & \vdots \\ B_0(\hat{S}_{L_{J-1},J-1}) & \cdots & B_M(\hat{S}_{L_{J-1},J-1}) \end{pmatrix} \tag{7.31}$$

最小化的解为：

$$a^{J-1} = ((B^{J-1})^T B^{J-1})^{-1} (B^{J-1})^T C^{J-1} \tag{7.32}$$

这样，获得存续价值 Cont^{J-1} 的向量为：

$$\mathrm{Cont}_i^{J-1} = \sum_{k=0}^{M} a_k^{J-1} B_{i,k}^{J-1} \tag{7.33}$$

一旦获得存续价值，我们就提前执行期权，当

$$P(\hat{S}_{i,J-1}) > \mathrm{Cont}_i^{J-1} \tag{7.34}$$

现金流向量 C^{J-1} 的元素 C_i^{J-1} 给定如下：

- 如果提前执行条件（7.34）被满足，则 $P(\hat{S}_{i,J-1})$.
- 否则为 0.

随后，对于条件（7.34）被满足的那些路径，现金流向量 C^J 的元素被设定为 0.

我们接着随时间向后回退，直到第一步.[24] 在每一个时间步长，正如前面所描述，提前执行被施行. 值得注意的是，在时间步长 t_k 时刻，每当现金流来自于路径 i 被提前执行，在这个路径中迟于 t_k 的所有现金流被剔除.[25]

最终，我们能够从现金流向量 C^k 建立现金流矩阵 C 以连接现金流向量 $C^k(k=1, \cdots, J)$，期权价值为各行之和的算术平均.[26]

7.5 扩展 Longstaff-Schwartz 至摆动期权

根据 Doerr 的陈述，[27] 我们展示了最小方差蒙特卡罗能被用于摆动期权的估值. 因为我们现在拥有不止一个执行权力，然而我们必须处理额外的“维数”，即剩余执行权力的数目. 考虑在

时刻 t_1、t_2、t_3、t_4 和 t_5 有执行机会的摆动期权，对于每一个机会 X，某一执行价格有 5 个执行机会和 3 个执行权利（向上摆动）. N 条路径生成 $N \times 5$ 现货价格矩阵 S. 不止一个执行权力带来的主要困难如下：

- 立即执行的好处不仅在于报酬，同样在于报酬加上剩余摆动期权的价值（摆动期权有一个向上摆动的权利，少于初始的摆动期权）.
- 在时刻 t_k 提前执行，在随后的机会重新分配现金流需要摆动期权的现金流矩阵，这个摆动期权有一个向上摆动的权利，少于初始的摆动期权.

因此，算法的一般形式的现金流矩阵必须为三维：

- 第一维：路径条数.
- 第二维：时间步长数目（执行机会）.
- 第三维：剩余执行权利（向上摆动）数据.

我们记剩余 j 个向上摆动的现金流矩阵为 C^j. 因此在我们的例子中，存在三个 $N \times 5$ 矩阵 C^1、C^2 和 C^3. 在初始步之后，现金流矩阵如下：

$$C^3 = \begin{bmatrix} \% & \% & P(S_{1,3}) & P(S_{1,4}) & P(S_{1,5}) \\ \% & \% & P(S_{2,3}) & P(S_{2,4}) & P(S_{2,5}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \% & \% & P(S_{N,3}) & P(S_{N,4}) & P(S_{N,5}) \end{bmatrix} \tag{7.35}$$

$$C^2 = \begin{bmatrix} \% & \% & \% & P(S_{1,4}) & P(S_{1,5}) \\ \% & \% & \% & P(S_{2,4}) & P(S_{2,5}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \% & \% & \% & P(S_{N,4}) & P(S_{N,5}) \end{bmatrix} \tag{7.36}$$

$$C^1 = \begin{bmatrix} \% & \% & \% & \% & P(S_{1,5}) \\ \% & \% & \% & \% & P(S_{2,5}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \% & \% & \% & \% & P(S_{N,5}) \end{bmatrix} \tag{7.37}$$

其中，

$$P(S) = \max(S - X, 0) \tag{7.38}$$

是向上摆动的报酬. %记号表示这些现金流在这个阶段是未定义的. 对于 C^3，我们能够结合算法的最后三个时间步长，因为显然提前执行发生在 t_3 时刻，而无论这个时间步长的报酬是否为正. 值得注意的是，这是从前一个开始的第三步.

类似地，当仅剩余两个向上摆动时，在 t_4 时刻立即提前执行，因此，我们能够结合 C^2 的最后两步. C^1 对应于百慕大期权 Longstaff-Schwartz 算法的现金流矩阵. 作为一个例子，如果我们仅有 6 条路径，在初始步之后，矩阵如下：

$$C^3 = \begin{bmatrix} \% & \% & P_{1,3} & 0 & P_{1,5} \\ \% & \% & P_{2,3} & P_{2,4} & 0 \\ \% & \% & P_{3,3} & 0 & 0 \\ \% & \% & P_{4,3} & P_{4,4} & P_{4,5} \\ \% & \% & P_{5,3} & 0 & P_{5,5} \\ \% & \% & P_{6,3} & P_{6,4} & P_{6,5} \end{bmatrix} \tag{7.39}$$

$$C^2 = \begin{bmatrix} \% & \% & \% & 0 & P_{1,5} \\ \% & \% & \% & P_{2,4} & 0 \\ \% & \% & \% & 0 & 0 \\ \% & \% & \% & P_{4,4} & P_{4,5} \\ \% & \% & \% & 0 & P_{5,5} \\ \% & \% & \% & P_{6,4} & P_{6,5} \end{bmatrix} \tag{7.40}$$

$$C^1 = \begin{bmatrix} \% & \% & \% & \% & P_{1,5} \\ \% & \% & \% & \% & 0 \\ \% & \% & \% & \% & 0 \\ \% & \% & \% & \% & P_{4,5} \\ \% & \% & \% & \% & P_{5,5} \\ \% & \% & \% & \% & P_{6,5} \end{bmatrix} \tag{7.41}$$

这里，所有 $P_{i,j}=P(S_{i,j})$ 都不等于零.

我们开始随着时间向后递推. 对于 t_4，我们运用现金流矩阵$\hat{C}_5^1$ 的最小方差回归到基函数上，计算仅剩余一个向上摆动期权的存续价值.[28] 在$\hat{C}_5^1$ 里，仅考虑在时刻 t_4 报酬是正的路径. 我们把在第 4 步剩余一个向上摆动的存续价值矩阵记为 Cont_4^1.

考虑到存续价值，我们可以实施提前执行权，C^1 为：

$$C^1 = \begin{bmatrix} \% & \% & \% & 0 & P_{1,5} \\ \% & \% & \% & P_{2,4} & 0 \\ \% & \% & \% & 0 & 0 \\ \% & \% & \% & P_{4,4} & 0 \\ \% & \% & \% & 0 & P_{5,5} \\ \% & \% & \% & P_{6,4} & 0 \end{bmatrix}$$

在我们的例子中，在时刻 t_4 所有路径（即路径 2、4 和 6）都被提前执行. 值得注意的是，对于路径 4 和 6，在时刻 t_5 的现金流被剔除. 在这一步中，现金流矩阵 C^2 和 C^3 仍然保持不变.

现在我们转向时刻 t_3. 为了取得 Cont_3^2，首先必须加入现金流向量 C_4^2 和 C_5^2. 和向量记为 C_{4+5}^2，通过忽略所有 $P(S_{i,3})$ 不等于 0 的路径，我们获得了相关和向量$\hat{C}_{4+5}^2$. 存续价值 Cont_3^2 可以通过$\hat{C}_{4+5}^2$在基函数的线性回归获得.

提前执行条件现在可以理解为：

$$P(S_{i,3}) + \mathrm{Cont}_3^1(i) > \mathrm{Cont}_3^2(i) \tag{7.42}$$

那意味着我们在 C^2 里提前执行前不得不计算 Cont_3^1. 根据常用的 Longstaff-Schwartz 算法，这个计算容易完成. 对于那些满足条件（7.42）的路径，提前执行被实施. 这意味着对于每一个相应的路径 i，$C_3^2(i)$ 等于报酬 $P(S_{i,3})$，现金流 $C_4^2(i)$ 和 $C_5^2(i)$ 各自被 $C_4^1(i)$ 和 $C_5^1(i)$ 替代. 之后，时刻 t_3 的提前执行在 C^1 中实施. 尽管 C^3 在这一步仍然保持不变，但是我们例子中的其他现金流矩阵可能如下：

$$C^2 = \begin{pmatrix} \% & \% & P_{1,3} & 0 & P_{1,5} \\ \% & \% & P_{2,3} & P_{2,4} & 0 \\ \% & \% & P_{3,3} & 0 & 0 \\ \% & \% & 0 & P_{4,4} & P_{4,5} \\ \% & \% & 0 & 0 & P_{5,5} \\ \% & \% & P_{6,3} & P_{6,4} & 0 \end{pmatrix} \tag{7.43}$$

$$C^1 = \begin{pmatrix} \% & \% & 0 & 0 & P_{1,5} \\ \% & \% & P_{2,3} & 0 & 0 \\ \% & \% & 0 & 0 & 0 \\ \% & \% & 0 & P_{4,4} & 0 \\ \% & \% & 0 & 0 & P_{5,5} \\ \% & \% & P_{6,3} & 0 & 0 \end{pmatrix} \tag{7.44}$$

对于 C^2，提前执行在路径 1，2，3 和 6 被实施. 值得注意的是，在路径 6 中，根据迭代步之前的 C^1，时刻 t_3 后的现金流须被修正. 对于 C^1，提前执行仅发生在路径 2 和 6.

转到时刻 t_2，对于三个现金流矩阵，提前执行都必须被实施. 首先，正如前所述，通过回归$\hat{C}^J_{3+4+5}$（$J=1，2，3$）到基函数，存续向量得以被计算出来. 然后，通过评估如下条件，始于 C^3 的提前执行被实施：

$$P(S_{i,2}) + \mathrm{Cont}_2^2(i) > \mathrm{Cont}_2^3(i) \tag{7.45}$$

对于 C^2 的条件为：

$$P(S_{i,2}) + \mathrm{Cont}_2^1(i) > \mathrm{Cont}_2^2(i) \tag{7.46}$$

对于 C^1，可得：

$$P(S_{i,2}) > \mathrm{Cont}_2^1(i) \tag{7.47}$$

因为提前执行包括根据先前迭代步现金流矩阵（比一个向上摆动要少），重新安排时刻 t_2 后的现金流，首先完成 C^3，然后是 C^2，最后是 C^1 这一程序是非常重要的. 对于 t_1 时刻重复同一个程序，我们结束于最终现金流矩阵 C^1、C^2 和 C^3. 从这些矩阵中，可以通过对各行和求平均获取对应于摆动期权的价值.

7.6 一般情形：向上摆动、向下摆动和惩罚函数

应用 LSM 算法来对摆动期权定价不仅必须结合看涨（向上摆动），而且必须结合看跌（向下摆动）. 另外，算法必须结合惩罚函数，而惩罚函数取决于执行的总数（向上摆动和向下摆动）.[30] 这些特性对于 LSM 算法有两个重要的意义. 第一，在初始步，因为惩罚的原因，存续价值必须是负的，仅仅最后一步需要处理. 因此，留下一个或多个执行权利到期日是没有价值的. 第二，我们有另外的维度，即剩余向下摆动的数目.

我们介绍一些符号：

- u 和 d 是各自被执行的向上摆动和向下摆动的数目（在特殊的迭代步）.
- $u_{\max}$ 和 $d_{\max}$ 是各自向上摆动和向下摆动的总数目.
- J 是时间步长的数目（执行机会）.

一般化的现金流张量有4维（路径、时间步长、被执行的向上摆动和被执行的向下摆动），是由 $[(u_{\max}+1)\cdot(d_{\max}+1)-1]$ 构成的现金流矩阵 $C^{u,d}$. 这些矩阵都是 $N\times J$ 的.

在初始步，我们不得不评估最后一个执行机会的现金流. 对于 u 个被执行的向上摆动和 d 个被执行的向下摆动，用 $\phi(u, d)$ 表示惩罚函数. 我们可以获得在最终时间步长 t_J 上路径 i 的现金流：对于 $0\leqslant u<u_{\max}$，$0\leqslant d<d_{\max}$，有：

$$C_J^{u,d}(i)=\max[P_u(S_{i,J})-\phi(u+1,d),P_d(S_{i,J})-\phi(u,d+1),0] \tag{7.48}$$

对于 $0\leqslant d<d_{\max}$，有：

$$C_J^{u_{\max},d}(i)=\max[P_d(S_{i,J})-\phi(u_{\max},d+1),0] \tag{7.49}$$

对于 $0\leqslant u<u_{\max}$，有：

$$C_J^{u,d_{\max}}(i)=\max[P_u(S_{i,J})-\phi(u+1,d_{\max}),0] \tag{7.50}$$

其中，$P_{u,d}$是向上摆动和向下摆动的报酬.

对于给定路径，现在开始从 t_J 时刻向后工作，不得不进行如下步骤：

- 计算最小方差回归的存续价值.
- 实施提前执行权.

当实施提前执行权时，必须从 $(u, d)=(0, 0)$ 步入 $(u, d)=(u_{\max}, d_{\max})$. 是否从 u 或者 d 开始并不重要. 在时间步长 j（因此迭代步 $J+1-j$），向上摆动的提前执行条件是对于 $0\leqslant u<u_{\max}$，$0\leqslant d<d_{\max}$，$u+d<j$，有：

$$P_u(S_{i,j})+\mathrm{Cont}_j^{u+1,d}(i)>\mathrm{Cont}_j^{u,d}(i) \tag{7.51}$$

对于向下摆动，可以获得对于 $0\leqslant u<u_{\max}$，$0\leqslant d<d_{\max}$，$u+d<j$，有：

$$P_d(S_{i,j})+\mathrm{Cont}_j^{u,d+1}(i)>\mathrm{Cont}_j^{u,d}(i) \tag{7.52}$$

事实上，在最终迭代步之后，摆动期权的价值可以通过计算 C_0^0 各行之和获得. 值得注意的是，在每一行中，至多有 $u_{\max}+d_{\max}$非零现金流.

7.7 应用 Matlab 定价摆动期权

参见附录B，可以看到由 Doerr(2004) 编写的代码完全列表. 对于使用向上摆动、向下摆动和惩罚函数定价摆动期权的 LSM 算法，代码提供了实现方法.

7.8 LSM 模拟结果

Doerr(2003) 使用单因素和双因素均值回复模型计算了摆动期权的价值. 表 7.2 给出的参数适用于单因素模型——参见公式 (7.2).

表 7.2

现货价格 S	20	每个采样率执行价格 K	20
采样率	10	均值回复速度 α	0.5
两个采样率之间间隔 δ_t	1	均值回复水平 F	20.738 7
执行权（向上摆动）	6	波动率 σ	0.392

资料来源：Doerr(2004).

图 7.5 表明，对于均值回复速度的两个不同价值，摆动期权的价值是现货价格的函数. 摆

动包括 6 个向上摆动和 10 个机会.

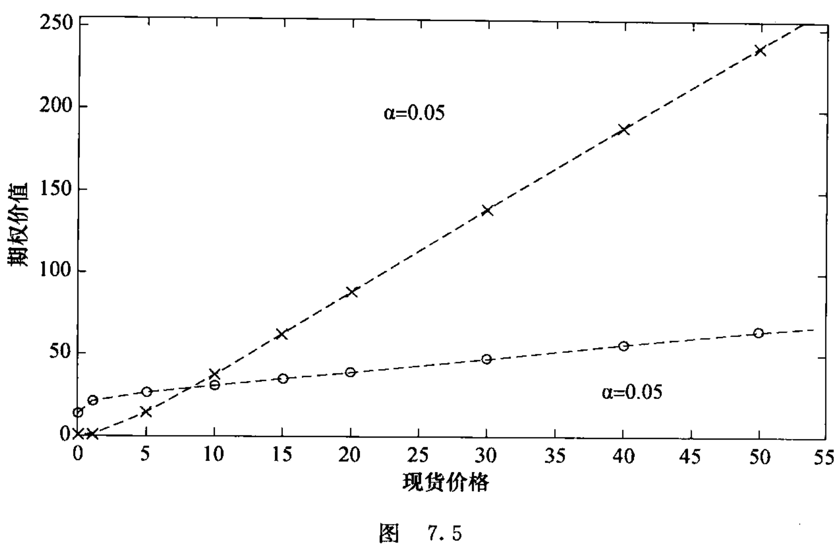


图 7.5
资料来源：Doerr(2003). 经允许后重印.

图 7.5 表明，对于有惩罚函数（左）和没有惩罚函数（右）均值回复速度的两个不同价值，摆动期权的价值是现货价格的函数. 摆动包括 6 个向上摆动和 10 个机会. 向下摆动的结果是期权价值最小. 引入一个惩罚：

$$\phi(v)=\begin{cases}50, & \text{若 } v\leqslant -1\\ 30(v-2), & \text{若 } v>2\\ 0, & \text{其他}\end{cases}$$

其中 $v=u-d$，u 和 d 各自为被执行的向上摆动和向下摆动的总数，这会降低期权价值，正如图 7.6 左半部分所示.

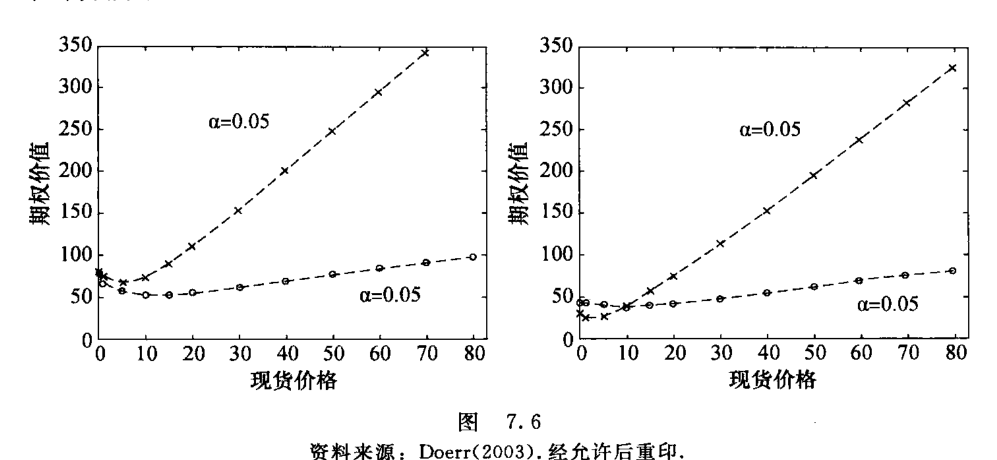


图 7.6
资料来源：Doerr(2003). 经允许后重印.

引入 4 个向下摆动期权和 6 个向上摆动期权（保持其他参数不变[31]）必然会导致期权价值的全面下降，因为如果保持老期权所有权利，[32] 则新期权拥有更多的执行权利，正如图 7.6 所

示. 因为对于较低的现货价格，向下摆动期权是价内的，对于所有的 α 值，期权价值作为现货价格的一个函数达到最小值.

图 7.7 显示了通过最小二乘蒙特卡罗和有限差分获得的摆动期权价值之间的相对差异. 通过赋予摆动期权 10 个机会，均值回复速度为 0.05(左边，没有向下摆动）和 0.5(右边，4 个向下摆动)，计算结果可以得出. 其他参数由表 7.2 给定.

相对偏差显著低于 1%，因此在计算精度之内. 对于蒙特卡罗，精度大约为 0.3%.

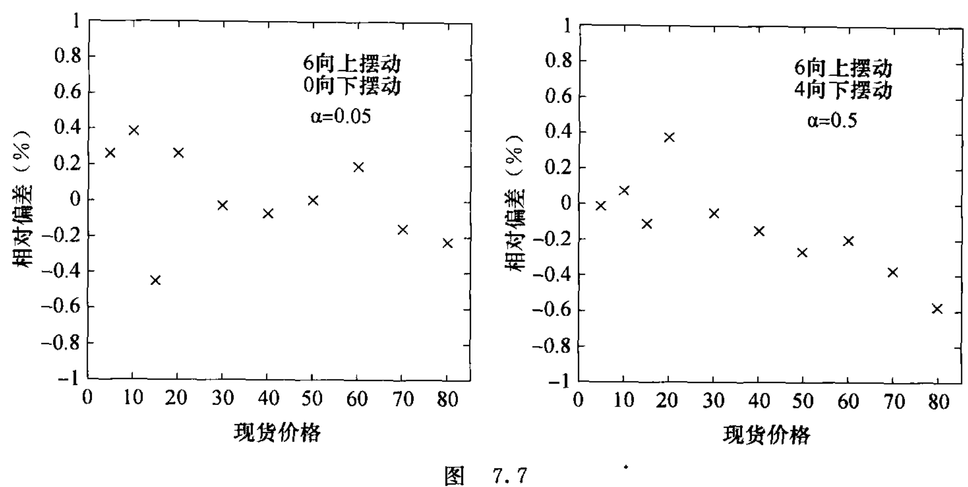


图 7.7

资料来源：Doerr(2003)，36.

7.8.1 上界与下界

一般情形下，因为摆动期权的计算价值成本相当高，我们总是试图找到尽可能简单的近似方法. 这里找到上界和下界作为近似方法的第一步是很重要的.

首先考虑简单情形，推导出拥有 m 个执行权利和 N 个机会的摆动期权的边界：

- 上界：m 个百慕大期权（根据摆动期权的机会，每一个拥有 N 个机会).
- 下界：Callstrip，即在 N 个单纯看涨期权集合中，m 最有价值的之和，其中摆动期权的 N 个机会失效.

这些边界经常作为文献讨论的主题（例如，参见 Jaillet、Ronn 和 Tompaidis[2003]），并能用如下方式解释：

- 上界：m 个百慕大期权的持有者能采取与摆动期权持有者同样的方式被执行. 进一步来说，同样机会下，他有执行 1 个以上期权的权利. 这意味着他比摆动期权的持有者有更多的可能性，因此 m 个百慕大期权的集合是摆动期权的上边界.
- 下界：摆动期权的持有者能在每一个机会执行，而 Callstrip 的持有者被限制于一开始就确定的 m 个执行日. 摆动期权持有者能在这 m 个日期被执行，但没必要. 因此，Callstrip 必须有一个下界. 下一步，我们想研究在这两个边界中何处摆动期权的价值是合适的. 因此，引入仓位 p：

$$p = \frac{\text{摆动期权的价值} - \text{下边界}}{\text{上边界} - \text{下边界}}$$

图 7.8 表明，对于由公式（7.24）给出的双因素均值回复过程，经验误差作为多条路径的一个函数，参数如表 7.3 所示.

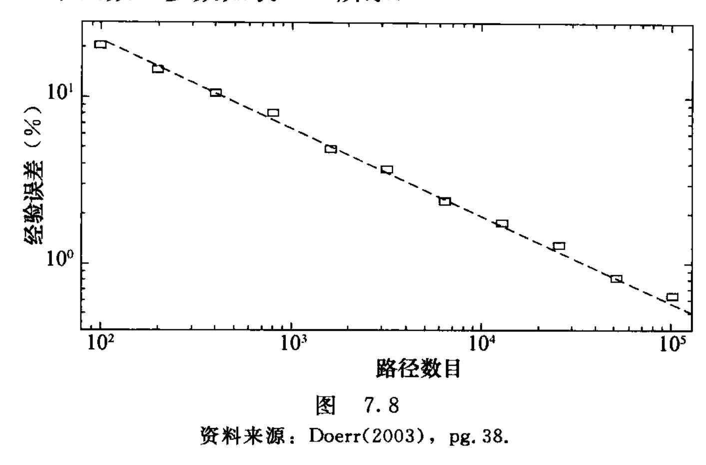


图 7.8

资料来源：Doerr(2003)，pg. 38.

表 7.3

现货价格 S	20
采样率	6
两个采样率之间间隔 δ_t	1
执行权（向上摆动）	4
每个采样率执行价格 K	20
α	0.5
δ_0	0.05
κ	0.03
σ_S	0.4
σ_δ	0.05
ρ	0.5
r	0.04

资料来源：Doerr(2003)，36.

正如单因素过程，图 7.8 中在对数坐标下出现一条近似直线.

显然，p 介于 0 和 1 之间. 对于固定的 N，考虑 p 作为 m 的一个函数，可立即找到两个平凡的情形：

$$p(1)=1 \tag{7.53}$$

$$p(N)=0 \tag{7.54}$$

因为对于 $m=1(N)$，摆动期权同百慕大期权（Callstrip）一样. 作为一次计算机实验，对于所有过程和各种过程参数集，$p(m)$ 都已经被决定.

式（7.53）和（7.54）的两种极限情形中的直线 u 给定如下：

$$u(m)=\frac{N-m}{N-1}$$

仓位 $u(m)$ 对应于摆动期权价值 $V_u(m)$：

$$V_u(m)=\frac{N-m}{N-1}m\cdot\text{百慕大}+\frac{m-1}{N-1}\cdot Callstrip$$

图 7.9 表明了双因素过程摆动期权的价值是现货价格的一个函数. 对于单因素过程，加入向下摆动期权导致总体的增加和最小值（中间值）的出现，随后惩罚函数的引入导致期权价值总体

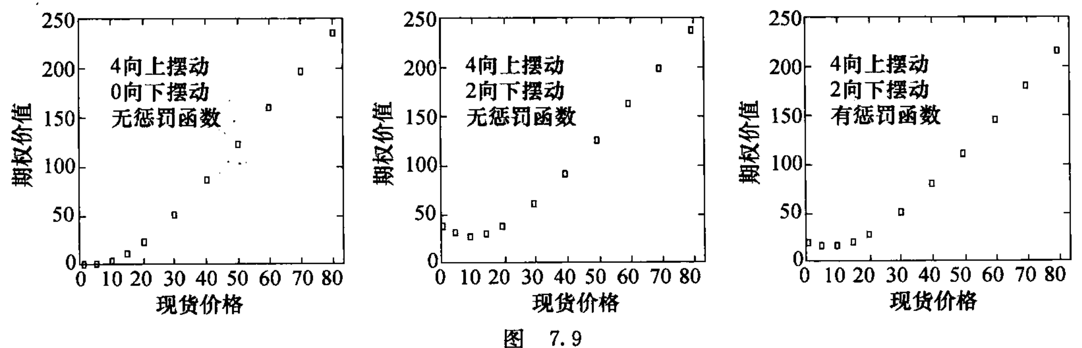


图 7.9

资料来源：Doerr(2003)，pg. 39.

的下降（右）.

7.8.2 执行策略

Doerr(2003) 讨论了影响摆动期权价值的各种执行策略.摆动期权价值是基于如下假设（隐含的）：

- 持有者使用最优执行策略.
- 提前执行的报酬能够立即实现.

然而，正如 Dörr 指出，这些假设在现实中并不完全需要.首先，运用最优执行策略需要每个机会的知识来判断执行好还是不执行好.其次，如果物理上的交割确定了，持有者也许不能从提前执行中获益，因为他必须卖出电力来交割.在这一部分中，我们关注各种不同执行策略下摆动期权的定价——即特殊执行策略被应用时，我们决定预期报酬.特别是，对如下持有者，我们试图发现一种方法来评估他们持有的摆动期权的价值：

- 并不明确地知道最优执行策略，或
- 执行时不能由自己确定，因为他已经被暴露在外部约束中.

第一种情形的目标就是找到一个简单的策略，这个策略能产生一个接近于最优执行价值的期权价值.这个策略中对过程参数作了简单假设.第二种情形的一个例子可能如下：持有者为了保护自己不受非常高的现货价格影响而买入摆动期权，但他事先并不知道他对电力的需求在何时.当持有者对电力没有需求的时候，他就不能从提前执行中实现回报，因为他无法卖出交割的电力.在这种情形下，这个持有者有兴趣知道他的预期回报与期权市场价格之差，这被认为是基于最优执行.我们把自己限定到 7.14 节所讨论的单因素（算法）过程.这样就使得过程的参数数目保持较少.特别是，这使得我们可以系统研究均值回复和波动率之间如何相互影响，而且提前执行问题的某些方面也可以被分析.

7.8.3 提前行权的阈值

在时刻 t_0，持有者需要决定是否行权.如果持有者决定不执行，期权就变成单纯看涨期权.因此，提前行权是最优的，如果实现的报酬远大于看涨期权的价值，即

$$C(S,t_0,t) > S - K$$

其中，S 和 K 分别表示现货价格和执行（strike）价格.接下来，保持执行价格 K、均值回复水平 F 和至到期日的时间 $t-t_0$ 不变，并考虑单纯看涨期权的价值作为现货价格 S、均值回复速度 α 和波动率 σ 的一个函数.忽略时间变量：

$$C(S,\alpha,\sigma) = A(t,t_0)e^{v(t,t_0)/2}N^0_{v(t,t_0)}(d_1) - KN^0_{v(t,t_0)}(d_2) \tag{7.55}$$

其中，A 和 v 给定如下：

$$A(t,t_0) = S(t_0)^{e^{-\alpha(t-t_0)}}F^{1-e^{-\alpha(t-t_0)}} \tag{7.56}$$

和

$$v(t,t_0) = \frac{\sigma^2}{2\alpha}(1-e^{-2\alpha(t-t_0)}) \tag{7.57}$$

给定这些参数后，d_1 和 d_2 可以被写为：

$$d_1 = \log\frac{A}{K} + v = d_2 + v$$

$N_b^a(\cdot)$ 是累积标准正态分布，可定义为

$$N_b^a(\cdot)=\int_{-\infty}^{y}\frac{1}{\sqrt{2\pi b}}\mathrm{e}^{-\frac{(x-a)^2}{2b}}\mathrm{d}x$$

对公式（7.55）关于 S 求偏导可得看涨期权的 delta：

$$\frac{\partial C}{\partial S}=\mathrm{e}^{v/2}\frac{\partial A}{\partial S}\left(N_v^0(d_1)+\frac{1}{\sqrt{2\pi v}}\mathrm{e}^{-\frac{d_1^2}{2v}}\right)-\frac{K}{A}\frac{\partial A}{\partial S}\frac{1}{\sqrt{2\pi v}}\mathrm{e}^{-\frac{d_2^2}{2v}}$$

其中

$$\frac{\partial A}{\partial S}=xS^{x-1}F^{1-x}$$

$$x=\mathrm{e}^{-\alpha(t-t_0)}$$

因为 $0<x<1$，当 $S\to\infty$ 时，可以获得如下极限：

$$A\to\infty$$

$$\frac{\partial A}{\partial S}\to 0$$

$$d_1\to\infty$$

$$d_2\to\infty$$

由此得出，它直接遵循：

$$\lim_{S\to\infty}\frac{\partial C}{\partial S}=0$$

图 7.10 展示了对应于现货价格看涨期权的典型曲线图以及立即执行的收益. 因为看涨期权总是大于 0，并且当 $S\to\infty$ 时，delta 趋近于 0，所以方程

$$C(X,t_0,t)=\max(X-K,0)$$

对于 X 总是有解，即存在一个提前执行（独特）阈值（threshold）.

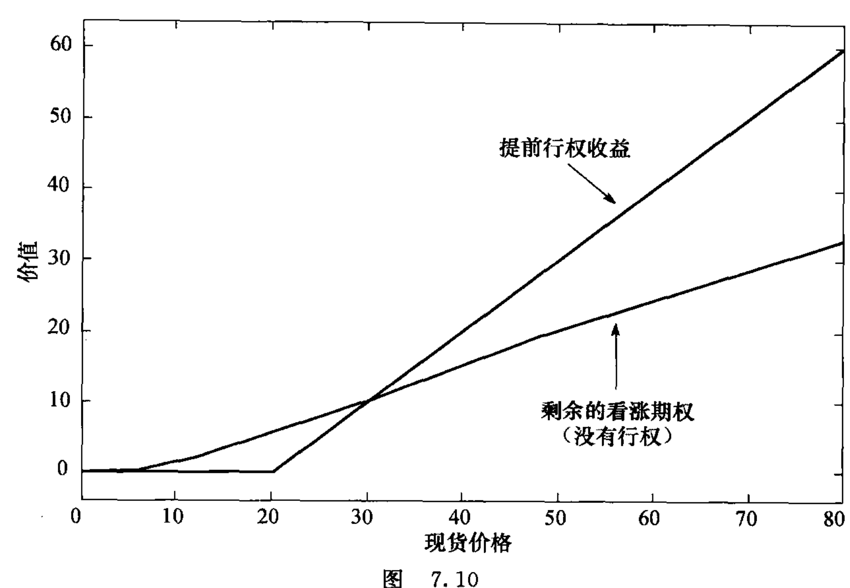


图 7.10

资料来源：Doerr(2003)，48.

我们能表明该阈值总是远大于均值回复水平 F. 直观上这是显然的——如果我们知道现货价格将趋近于 F，为什么要提前执行？

Doerr(2003) 发现了提前执行的阈值：

- 当 $\alpha>0$ 时总是存在.
- 总是远大于均值回复的水平.
- 随着 α 的增加而减小，当 $\alpha\to\infty$ 时接近于 f.
- 随着 σ 增加而增大.

这些结果如图 7.11 所示. 直观上，随着均值回复速度的增加，阈值（总是处于均值水平之上）下降并不令人惊奇. 如果 α 较大，我们应该立即使用提前执行权，因为现货价格预期将会被均值回复水平拉下来. 然而，如果 α 较大，当显著高于均值回复速度时，仍然有一些人希望现货价格会上升. 因此，提前执行的需求不那么迫切.

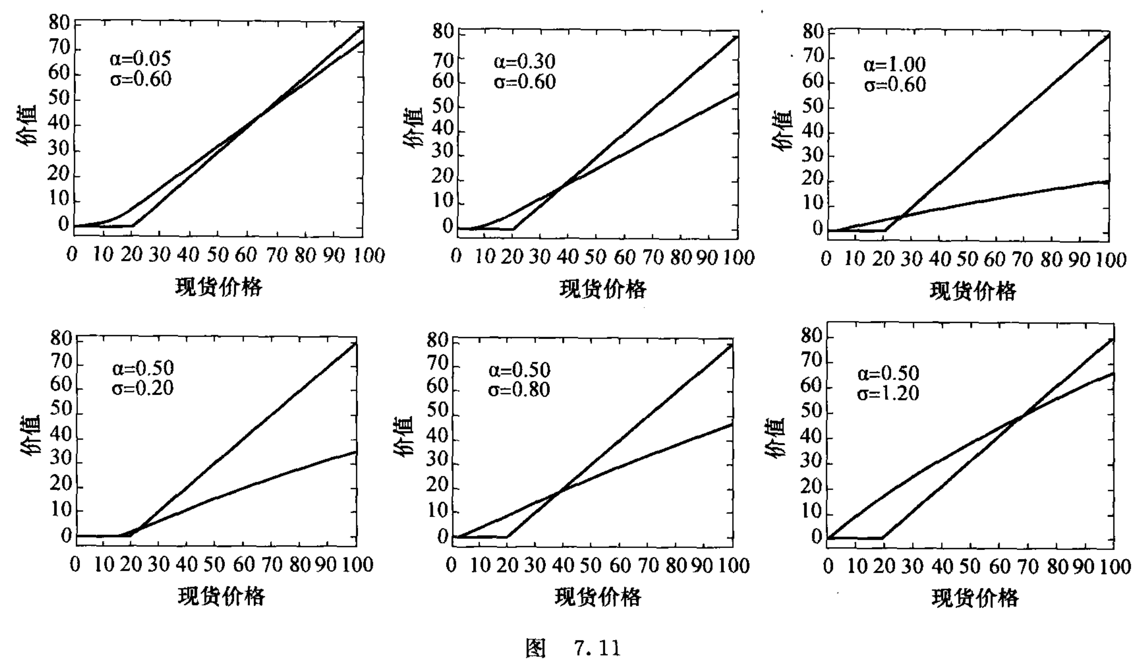


图 7.11

资料来源：Doerr(2003)，51.

7.8.4 提前执行和期权价值的相互影响

对于百慕大期权，持有者通过提前执行来最大化它的预期收益. 最优执行策略是相当简单的：如果在时刻 t_0 的现货价格超过阈值 X，则决定执行. 然而，应用最优策略需要关于 X 的确切知识. 在简单的情形中，非常容易从数值上决定 X，但是对于更现实的情形（例如，多于一个机会），情况并不是这样. 因此，更详细地研究提前执行的好处非常重要. 这意味着我们需要知道实际策略中预期期权收益的敏感性. 现在，这从直观上应该可以实现. 如图 7.11 所示，提前执行收益与存续函数的夹角随着 α 增加而增大. 在图 7.12 中，这显示的更为明显. 对于 $\alpha=0.05$，曲线非常接近于其他曲线，次最优提前执行决定预计对于期望的期

权收益没有实际影响. 然而，当 $\alpha=0.5$ 则相反. 在此种情形下，关于实际策略预期收益的敏感性必然是显著的.

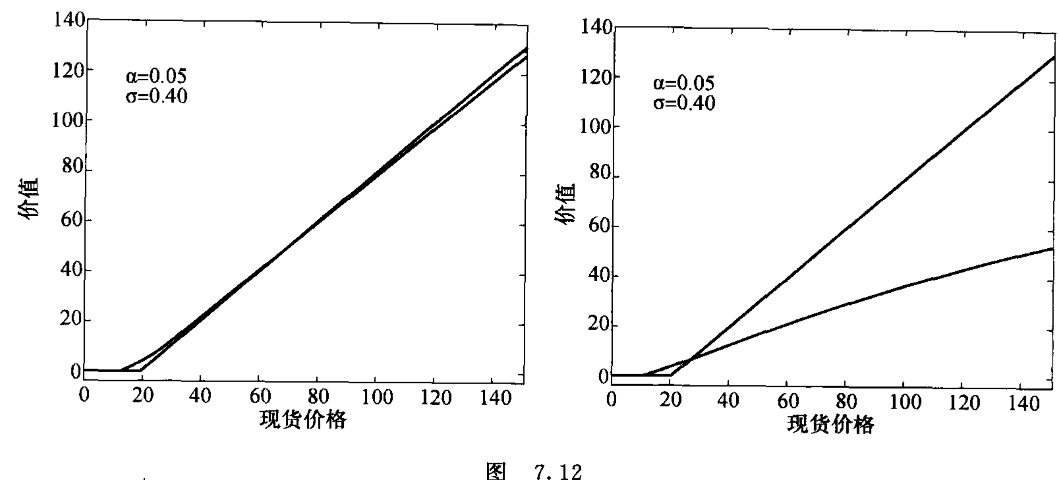


图 7.12

资料来源：Doerr(2003), 52.

夹角 ϕ 给定如下：

$$\phi(\alpha,\sigma) = \frac{\pi}{4} - \arctan\left(\frac{\partial C}{\partial S}\bigg|_{S=X}\right)$$

我们可以使用 ϕ 作为策略预期期权收益敏感性的指标. 对于如前所述简单的情形，显然敏感性随着 ϕ 的增加而增大. 因为 ϕ 的分析处理是非常困难的（如果可能），但可以用数值方法解决. 随着 α 的增加，ϕ 也增大，而 ϕ 随着波动率的增加而减少. 然而，与 α 的相关性似乎比 σ 要大得多. 更多的细节参见 Doerr(2003).

7.9 能源商品衍生品的定价

7.9.1 跨商品价差期权

在石油产业中，精炼企业的管理者更关注他们的输入和产出价格之差，而不是价格水平. 精炼企业的利润直接与价差相关，或是与原油价格和精炼产品的价格差有关. 因为提炼者能可靠地预测他们的成本，而不是原油，所以价差成为他们主要的不确定性因素. 原油市场的裂解价差期权与电力市场点火价差期权和地域价差期权一样是跨商品衍生品的典型例子，它们在风险控制中起着至关重要的作用. 点火价差期权是电力市场的衍生品，矿物燃料被用来发电. 这样一种期权在矿物燃料发电厂资产评估中是必不可少的.[33] 欧式点火价差看涨（SSC）期权支付电力现货价格与到期日生产燃料成本之差的正值部分. 其收益函数为：

$$SSC(S_t^e, S_T^g, H, T) = \max(S_T^e - H \cdot S_T^g, 0) \tag{7.58}$$

其中，S_T^e 和 S_T^g 分别表示电力价格和生产燃料价格. 常数 H 是执行热价（strike heat price），它代表按照合约发出单位电量所需的生产燃料的数量.

一个区域价差期权支付了两个不同交割点标的商品价格正的价格差价. 在电力市场中，区

域价差期权是为了对冲传输风险而存在的，它也能被用来对传输扩展工程估值.[34]（参见 Deng、Johnson 和 Songomonia[1998]）. 欧式区域价差看涨期权（LSC）的收益为：

$$LSC(S_T^A, S_T^B, L, T) = \max(S_T^B - L \cdot S_T^A, 0) \tag{7.59}$$

其中，S_T^a 和 S_T^b 是当地 A 和 B 的商品价格. 常数 L 是 "反映运输或传输损失的亏损因子或是从 A 地运输一个单位商品到 B 地的成本."[35]

普通的跨商品价差看涨期权（CSC）是在到期日 T 拥有如下收益的期权：

$$CSC(S_T^1, S_T^2, K, T) = \max(S_T^1 - K \cdot S_T^2, 0) \tag{7.60}$$

其中，S_T^i 是商品 $i(i=1, 2)$ 的现货价格，K 是第二种商品现货价格相关的度量常数. 不同跨商品期权 K 的解释是不一样的. 例如，K 表示点火价差期权的执行热率 H，而在区域价差期权中代表亏损因子 L.

在各种提出的模型中，标的能源商品的欧式未定权益（contingent claim）的价格能通过特征转换函数的求逆获得. 假定 $\boldsymbol{X}_t$ 是 $\mathcal{R}^n$ 的一个状态变量，$u \in C^n$. 一般化的特征转换函数可定义为：

$$\varphi(u, \boldsymbol{X}_t, t, T) = E^Q[\mathrm{e}^{-r(T-t)} \exp(u \cdot \boldsymbol{X}_T) | \Im_t] = \exp[A(t,u) + B(t,u)\boldsymbol{X}_t]$$

用 $\varphi(u, \boldsymbol{X}_t, t, T)$ 标记在时刻 t 未定权益价格的报酬 $\exp(\boldsymbol{a} \cdot \boldsymbol{X}_T)$，当在时刻 T 满足 $\boldsymbol{b} \cdot \boldsymbol{X}_T \leqslant v$ 时，其中 $\boldsymbol{a}$，$\boldsymbol{b}$ 是在 $\mathcal{R}^n$ 中的向量，$v \in \mathcal{R}^1$. 接下来有：

$$\begin{aligned} G(v, \boldsymbol{X}_t, t, T; \boldsymbol{a}, \boldsymbol{b}) &= E^Q \lfloor \mathrm{e}^{-r(T-t)} \exp(\boldsymbol{a} \cdot \boldsymbol{X}_t) 1_{\boldsymbol{b} \cdot \boldsymbol{X}_T \leqslant v} | \Im_t \rfloor \\ &= \frac{\varphi(\boldsymbol{a}, \boldsymbol{X}_t, t, T)}{2} - \frac{1}{\pi} \int_0^\infty \frac{\mathrm{Im}\{\varphi(\boldsymbol{a} + iw\boldsymbol{b}, \boldsymbol{X}_t, t, T)\mathrm{e}^{-iwv}\}}{w} \mathrm{d}w \end{aligned}$$

如果 v，$\boldsymbol{a}$ 和 $\boldsymbol{b}$ 的选择合适，$G(v, \boldsymbol{X}_t, t, T; \boldsymbol{a}, \boldsymbol{b})$ 作为建立的模块定价未定权益，如远期/期货、看涨/看跌期权和跨商品价差期权.

两个商品的欧式跨商品价差看涨期权的价值能通过式（7.58）的傅里叶变换找到. 它的价格给定如下：

$$\begin{aligned} CSC(S_t^1, S_t^2, K, t) &= E^Q \lfloor \mathrm{e}^{-r(T-t)} \max(S_T^1 - K \cdot S_T^2, 0 | \Im_t \rfloor \\ &= E^Q \lfloor \mathrm{e}^{-r(T-t)} \exp(X_T^1) 1_{S_T^1 - K \cdot S_T^2 \geqslant 0} | \Im_t \rfloor - K \cdot E^Q \lfloor \mathrm{e}^{-r(T-t)} \exp(X_T^2) 1_{S_T^1 - K \cdot S_T^2 \geqslant 0} | \Im_t \rfloor \\ &= G_1 - K \cdot G_2 \end{aligned}$$

其中，

$$\begin{aligned} G_1 &= G(0, \ln S_t^1, \ln(K \cdot S_t^2), t, T; [1, 0, \cdots, 0]'; [-1, 1, 0, \cdots, 0]') \\ G_2 &= G(0, \ln S_t^1, \ln(K \cdot S_t^2), t, T; [0, 1, \cdots, 0]'; [-1, 1, 0, \cdots, 0]') \end{aligned} \tag{7.61}$$

和

$$G(v, \boldsymbol{X}_t, t, T; \boldsymbol{a}, \boldsymbol{b}) = \frac{\varphi(\boldsymbol{a}, \boldsymbol{X}_t, t, T)}{2} - \frac{1}{\pi} \int_0^\infty \frac{\mathrm{Im}\{\varphi(\boldsymbol{a} + iw\boldsymbol{b}, \boldsymbol{X}_t, t, T)\mathrm{e}^{-iwv}\}}{w} \mathrm{d}w$$

是在时刻 t 未定权益的价格：

$$\varphi(u, X_t, Y_t, t, T) = E^Q[\mathrm{e}^{-r(T-t)} \exp(u_1 X_T + u_2 Y_T) | \Im_t]$$

为转换函数的一般形式，其中 $\boldsymbol{a}$，$\boldsymbol{b}$ 是在 $\mathcal{R}^n$ 中的向量，$v \in \mathcal{R}^1$（对于偏差参见 Duffie、Pan 和 Singleton[1998]）.

为了定价能源可交易商品衍生品，Deng(1999) 考虑了三个一般化的模型：结构转换、确

定波动率跳跃-扩散和随机波动率跳跃-扩散.在每一个模型中，参数被假定为常数，跳跃仅出现在初级商品价格和波动过程（模型 3）.更重要的是，跳跃的大小服从$\mathcal{R}^n$空间内独立指数随机变量分布，因此有如下的转换函数：

$$\phi_J^j(c,t) = \prod_{k=1}^{n} \frac{1}{1-\mu_j^k c_k} \tag{7.62}$$

其中，c 为复数的向量.

7.9.2 模型 1

跳跃是初级商品现货价格的对数 X_t. j 类型跳跃的大小是均值为 μ_J^j 的指数分布.跳跃大小分布的转换函数是 $\phi_J^j(c_1, c_2, t)=\dfrac{1}{1-\mu_J^j c_1}$，$j=1$，2.

$$\mathrm{d}\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} \kappa_1(\theta_1 - X_t) \\ \kappa_2(\theta_2 - Y_t) \end{pmatrix}\mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ \rho_1\sigma_2 & \sqrt{1-\rho_1^2}\sigma_2 \end{bmatrix}\mathrm{d}W_t + \sum_{i=1}^{2}\Delta Z_t^i \tag{7.63}$$

对于这个模型，转换函数的闭型解能被显式地写为：

$$\varphi_1(u,X_t,Y_t,t,T) = \exp(\alpha(\tau) + \beta_1(\tau)X_t + \beta_2(\tau)Y_t)$$

其中，$\tau = T-t$.可以得出：[36]

$$\beta_1(\tau,u_1) = u_1 \mathrm{e}^{-\kappa_1\tau}$$

$$\beta_2(\tau,u_1) = u_2 \mathrm{e}^{-\kappa_2\tau}$$

$$\begin{aligned}\alpha(\tau,u) = &- r\tau - \sum_{j=1}^{2}\frac{\lambda_J^j}{\kappa_1}\ln\frac{u_1 u_J^j - 1}{u_1 u_J^j \mathrm{e}^{-\kappa_1\tau} - 1} + \frac{a_1\sigma_1^2 u_1^2}{4\kappa_1} + \frac{a_2\sigma_2^2 u_2^2}{4\kappa_2} + u_1\theta_1(1-\mathrm{e}^{-\kappa_1\tau}) \\ &+ u_2\theta_2(1-\mathrm{e}^{-\kappa_2\tau}) + \frac{u_1 u_2\rho_1\sigma_1\sigma_2(1-\mathrm{e}^{-(\kappa_1+\kappa_2)\tau})}{\kappa_1+\kappa_2}\end{aligned}$$

其中，$a_1 = 1-\mathrm{e}^{-2\kappa_1\tau}$，$a_2 = 1-\mathrm{e}^{-2\kappa_2\tau}$.

7.9.3 模型 2

模型 2 是有结构跳跃的结构转换模型，这种结构跳跃主要出现在初级商品价格过程中.在电力市场中，这很适合对初级商品随机价格尖峰建模，随机价格尖峰主要由发电厂或运输网络的线路偶然故障导致.这个模型可以用作电力联合规范和在风险中性测度 Q 下的燃油价格过程.为简便起见，Deng(1999) 假定每一个结构中都没有跳跃：

$$\mathrm{d}\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} \kappa_1(\theta_1 - X_t) \\ \kappa_2(\theta_2 - Y_t) \end{pmatrix}\mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ \rho_1\sigma_2 & \sqrt{1-\rho_1^2}\sigma_2 \end{bmatrix}\mathrm{d}W_t + v(U_{t-})\mathrm{d}M_t \tag{7.64}$$

U_t 是由连续时间双状态马尔可夫过程定义的结构状态过程：

$$\mathrm{d}U_t = 1_{U_{t=0}} \cdot \delta(U_t)\mathrm{d}N_t^0 + 1_{U_{t=1}} \cdot \delta(U_t)\mathrm{d}N_t^1 \tag{7.65}$$

其中，N_t^i 是到达密度为 $\lambda^i(i=0,\ 1)$ 的泊松过程，$\delta(0) = -\delta(1) = 1$. M_t 被定义为连续的马尔可夫链：

$$\mathrm{d}M_t = -\lambda(U_t)\delta(U_t)\mathrm{d}t + \mathrm{d}U_t \tag{7.66}$$

W_t 是标准布朗运动. $\{v(i) = (v_1(i),\ v_2(i))',\ i=0,\ 1\}$ 表示当结构转换发生时，状态变量随

机跳跃的大小.

$\phi_{v(i)}(c_1,c_2,t)=\int_{\mathcal{R}^2}\exp(c\cdot z)\mathrm{d}v_{v(i)}(z)$ 是结构跳跃大小分布 $v(i)(i=1,\ 2)$ 的转换函数. Z^j，ΔZ^j 和 ϕ^i_j 与在模型 1 中的定义相似.

模型的转换函数在闭合情形中是无法完全求解的. 我们得出：

$$\varphi_2^0(x,y,t)=\exp(\alpha_0(t)+\beta_1(t)x+\beta_2(t)y)$$

$$\varphi_2^1(x,y,t)=\exp(\alpha_0(t)+\beta_1(t)x+\beta_2(t)y)$$

其中，$\beta(t)=\beta(t,\ u)=(\beta_1(t,\ u),\ \beta_2(t,\ u))'$有闭型解：

$$\beta_1(\tau,u_1)=u_1\exp(-\kappa_1\tau)$$

$$\beta_2(\tau,u_1)=u_1\exp(-\kappa_2\tau)$$

$\alpha(t)=\alpha(t,\ u)=(\alpha_0(t,\ u),\ \alpha_1(t,\ u))'$需要从以下式子数值计算得出：

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\alpha_0(t)\\ \alpha_1(t)\end{pmatrix}=-\begin{bmatrix}A_1(\beta(t),t)+\lambda^0\left[\dfrac{\mathrm{e}^{(\alpha_1(t)-\alpha_0(t))}}{1-\mu_0\beta_1(t,u_1)}-1\right]\\ A_1(\beta(t),t)+\lambda^1\left[\dfrac{\mathrm{e}^{(\alpha_1(t)-\alpha_1(t))}}{1-\mu_1\beta_1(t,u_1)}-1\right]\end{bmatrix}$$

$$\begin{bmatrix}\alpha_0(0,u)\\ \alpha_1(0,u)\end{bmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}$$

其中

$$A_1(\beta(t),t)=-r+\sum_{i=1}^{2}\left[\kappa_i\theta_i\beta_i+\frac{1}{2}\sigma_i^2\beta_i^2\right]-\rho_1\sigma_1\sigma_2\beta_1\beta_2$$

7.9.4 模型 3

模型 3 是随机波动模型，第一类跳跃是商品现货价格和波动率过程的同时跳跃，第二类跳跃仅仅是商品现货价格. 所有参数为常数.

$$\mathrm{d}\begin{bmatrix}X_t\\ V_t\\ Y_t\end{bmatrix}=\begin{bmatrix}\kappa_1(\theta_1-X_t)\\ \kappa_V(\theta_V-V_t)\\ \kappa_2(\theta_2-Y_t)\end{bmatrix}\mathrm{d}t+\begin{bmatrix}\sqrt{V_t} & 0 & 0\\ \rho_1\sigma_2\sqrt{V_t} & \sqrt{(1-\rho_1^2)V_t\sigma^2} & 0\\ \rho_2\sigma_3\sqrt{V_t} & 0 & \sigma_3\end{bmatrix}\mathrm{d}W_t+\sum_{i=1}^{2}\Delta Z_t^i \tag{7.67}$$

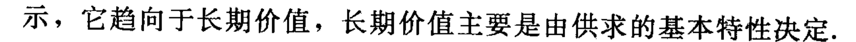
其中，W_t 是$\mathcal{R}^3$ 中的标准布朗运动. $Z^i(i=1,\ 2)$ 是$\mathcal{R}^3$ 中的复合泊松运动. 泊松到达密度函数为 $\lambda^1(X_t,\ V_t,\ Y_t,\ t)=\lambda_1$ 和 $\lambda^2(X_t,\ V_t,\ Y_t,\ t)=\lambda_2 V_t$. 跳跃大小分布的转换函数如下：

$$\phi_J^1(c_1,c_2,c_3,t)=\frac{1}{(1-\mu_1^1c_1)(1-\mu_1^2c_2)}$$

其中，μ_J^k 是因素 $k(k=1,\ 2)$ 中第 $J(J=1,\ 2)$ 类跳跃的均值大小. 转换函数的形式如下（参见 Deng[1999]）：

$$\varphi_3(u,X_t,V_t,Y_t,t,T)=\exp(\alpha(t,u)+\beta_1(t,u)X_T+\beta_2(t,u)V_T+\beta_3(t,u)Y_T)$$

对于前面三个模型和几何布朗运动（GBM）模型，Deng 使用这些模型对 $H=9.5$MMBtu/Mwh 的尖峰热率的点火价差看涨期权进行了定价. 点火价差看涨期权价值趋向于在 GBM 价格模型下的当前现货价格. 然而，在均值回复跳扩散价格模型中，正如图 7.13 所



示，它趋向于长期价值，长期价值主要是由供求的基本特性决定.

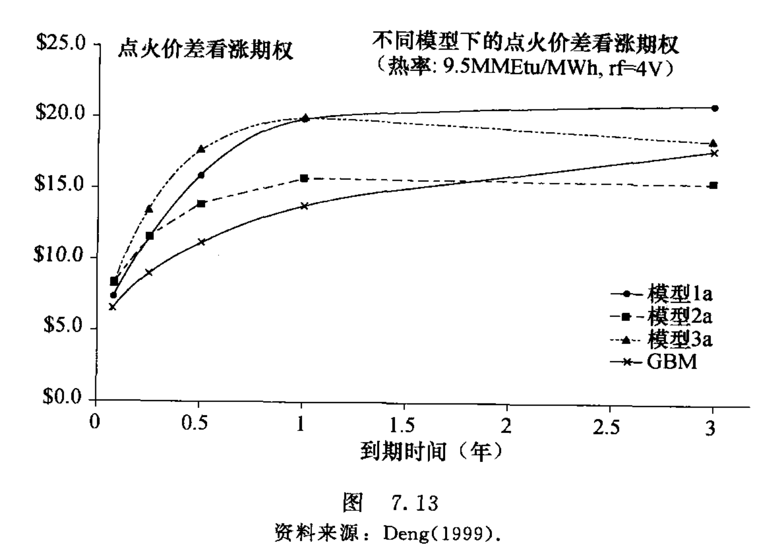


图 7.13

资料来源：Deng(1999).

Deng 使用了如表 7.4 所示的模型参数估计. 为了估计参数，Deng(1999) 从标的价格回报率非条件分布的转换函数得出了矩条件. Deng 假定与风险报酬相关的因素 X 与 X 成比例，例如，风险报酬的形式是 $\xi_X \cdot X$. 简单来说，Deng 假定与跳跃相关的风险报酬是零. 接着，根据真实观测和匹配矩条件与期货价格的风险报酬，Deng 使用了电力和天然气现货价格和期货价格序列来估计模型参数.[37]

表 7.4

	模型 1a	模型 2a	模型 3a		模型 1a	模型 2a	模型 3a
κ_1	1.70	1.37	2.17	σ_3	N/A	N/A	0.54
κ_2	1.80	1.80	3.50	ρ_1	0.20	0.20	0.25
κ_3	N/A	N/A	1.80	ρ_2	N/A	N/A	0.20
θ_1	3.40	3.30	3.20	λ_1	6.08	6.42	6.43
θ_2	0.087	0.87	0.85	μ_{11}	6.19	0.26	0.23
θ_3	N/A	N/A	0.87	μ_{12}	N/A	N/A	0.22
σ_1	0.74	0.80	N/A	λ_2	7.00	8.20	5.00
σ_2	0.34	0.34	0.80	μ_{21}	−0.11	−0.20	−0.14

资料来源：Deng(1999).

7.10 跳跃扩散型定价模型

为捕捉电力价格的均值回复和尖峰特性，提出了各种随机模型. 仿射跳跃扩散过程是普遍使用的一种，因为它们“弹性足够来捕捉一些属性，如多次跳跃、滚动时间长期均值和各式各样的随机波动率，这些属性在不牺牲计算方便性的基础上出现在许多金融时间序列中.”[38] 根据 Xiong(2004)，我们有下面几节中的模型.

7.10.1 模型 1a：仿射均值回复跳跃扩散过程

扩散部分由 Ornstein-Uhlenbeck 过程表示，具有由伯努利变量决定的跳跃符号的跳跃构成是跳跃大小的指数分布绝对值. 公式如下：

$$dX_t = \kappa(\alpha - X_t)dt + \sigma dW_t + J_t dP_t \tag{7.68}$$

其中，元组 $\theta=[k, \alpha, \sigma^2, \omega, \psi, \lambda]$ 是未知参数. 特别地，k 是均值回复速率，α 是长期均值，W_t 是 dW_t 的标准布朗运动，对于无穷小时间间隔 dt 服从 $N(0, dt)$，P_t 是到达率为 ω 的离散的一维标准泊松过程. 在 dt 中，如果有跳跃，则 $dP_t=1$，否则 $dP_t=0$. 跳跃振幅 J_t 是均值为 λ 的指数分布，跳跃 J_t 的符号是伯努利分布的，参数 ψ 是随机变量. 假定布朗运动、泊松过程和随机跳跃振幅都是马尔可夫过程并且两两独立.

给定 X_t，$\phi(s, \theta, X_T|X_t)$，X_T 的条件特征函数（CCF）为：

$$\begin{aligned}\varphi(s,\theta,X_T|X_t) &= E[\exp(isX_T)|X_t] \\ &= \exp(A(s,t,T,\theta) + B(s,t,T,\theta)X_t)\end{aligned} \tag{7.69}$$

其中，$A(\cdot)$ 和 $B(\cdot)$ 满足下列复值常微分方程（ODE）系统：

$$\begin{aligned}\frac{\partial A(s,t,T,\theta)}{\partial t} &= -\kappa\alpha B(s,t,T,\theta) \\ &\quad - \frac{1}{2}\sigma^2(s,t,T,\theta) - \omega(\varphi(B(s,t,T,\theta)) - 1), \\ \frac{\partial B(s,t,T,\theta)}{\partial t} &= \kappa B(s,t,T,\theta)\end{aligned} \tag{7.70}$$

其边界条件为：

$$A(s,T,T,\theta) = 0, \qquad B(s,T,T,\theta) = is \tag{7.71}$$

这里，“跳跃转换” $\varphi(B(s, t, T, \theta))$ 给定如下：

$$\begin{aligned}\varphi(B(s,t,T,\theta)) &= \psi\int_0^\infty \exp(B(s,t,T,\theta)z)\frac{1}{\gamma}\exp\left(-\frac{z}{\gamma}\right)dz \\ &\quad + (1-\psi)\int_0^\infty \exp(-B(s,t,T,\theta)z)\frac{1}{\gamma}\exp\left(-\frac{z}{\gamma}\right)dz \\ &= \frac{\psi}{1-B(s,t,T,\theta)\gamma} + \frac{1-\psi}{1+B(s,t,T,\theta)\gamma}\end{aligned} \tag{7.72}$$

对 $A(\cdot)$ 和 $B(\cdot)$ 求解式（7.69），并应用相应的边界条件得到如下结果：

$$\begin{aligned}A(s,t,T,\theta) &= i\alpha s(1-e^{-\kappa(T-t)}) - \frac{\sigma^2 s^2}{4\kappa}(1-e^{-2\kappa(T-t)}) \\ &\quad + \frac{i\omega(1-2\psi)}{\kappa}(\arctan(\gamma s e^{-\kappa(T-t)}) - \arctan(\gamma s)) \\ &\quad + \frac{\overline{\omega}}{2\kappa}\ln\left(\frac{1+\gamma^2 s^2 e^{-2\kappa(T-t)}}{1+\gamma^2 s^2}\right) \\ B(s,t,T,\theta) &= is e^{-\kappa(T-t)}\end{aligned} \tag{7.73}$$

7.10.2 模型 1b

这个模型考虑到了非对称上摆和下摆跳跃（参见 Deng[1999]，前面已讨论），每一个模型

有指数分布跳跃大小.现货价格 X_t 的对数满足 SDE:

$$dX_t = \kappa(\alpha - X_t)dt + \sigma dW_t + J_t^u dP_t^u(\omega) + J_t^d dP_t^d(\omega) \tag{7.74}$$

其中，κ 是均值回复速率，α 是长期均值，W_t 是满足 $dW_t N(0, dt)$ 的标准布朗运动. X_t 的跳跃行为取决于两种跳跃：向上跳跃和向下跳跃.向上跳跃 J_t^u 是正均值 γ_u 和跳跃到达速率 ω_u 的指数分布.向下跳跃 J_t^d 是负均值 γ_d 和跳跃到达速率 ω_d 的指数分布.反过来，P_t^u 和 P_t^d 各自是到达速率 ω_u 和 ω_d 的两个独立离散的一维标准泊松过程.

转换条件特征函数（CCF）可以写为：

$$\varphi(s,\theta,X_T|X_t) = E[\exp(isX_T)|X_t] = \exp(A(s,t,T,\theta) + B(s,t,T,\theta)X_t)$$

其中，$A(\cdot)$ 和 $B(\cdot)$ 满足复值常微分方程系统：

$$\begin{aligned}\frac{\partial A(s,t,T,\theta)}{\partial t} &= -\kappa\alpha B(s,t,T,\theta) - \frac{1}{2}\sigma^2(s,t,T,\theta) - \omega_u(\varphi_u(B(s,t,T,\theta)) - 1)\\ &\quad - \omega_d(\varphi_d(B(s,t,T,\theta)) - 1)\\ \frac{\partial B(s,t,T,\theta)}{\partial t} &= \kappa B(s,t,T,\theta)\end{aligned} \tag{7.75}$$

其边界条件为：

$$A(s,T,T,\theta) = 0, \qquad B(s,T,T,\theta) = is$$

这里，向上跳跃的“跳跃转换”给定如下：

$$\begin{aligned}\varphi_u(B(s,t,T,\theta)) &= \int_0^\infty \exp(B(s,t,T,\theta)z)\left[\frac{1}{\gamma_u}\exp\left(-\frac{z}{\gamma_u}\right)\right]dz\\ &= \frac{1}{1 - B(s,t,T,\theta)\gamma_u}\end{aligned}$$

类似地，向下跳跃的“跳跃转换”给定如下：

$$\varphi_d(B(s,t,T,\theta)) = \frac{1}{1 - B(s,t,T,\theta)\gamma_d}$$

经过一些计算，可以求出 $A(\cdot)$ 和 $B(\cdot)$ 的解，应用边界条件：

$$\begin{aligned}A(s,t,T,\theta) &= i\alpha s(1 - e^{-\kappa(T-t)}) - \frac{\sigma^2 s^2}{4\kappa}(1 - e^{-2\kappa(T-t)})\\ &\quad + \frac{\omega_u}{\kappa}\ln\left(\frac{1 - is\gamma_u e^{-\kappa(T-t)}}{1 - is\gamma_u}\right)\\ &\quad + \frac{\omega_d}{\kappa}\ln\left(\frac{1 - is\gamma_u e^{-\kappa(T-t)}}{1 - is\gamma_d}\right)\\ B(s,t,T,\theta) &= is e^{-\kappa(T-t)}.\end{aligned} \tag{7.76}$$

7.10.3 模型 2a：时变漂移项

通过用确定性函数 $\alpha(t)$ 替代长期均值 α，模型 1(7.68) 可通过加入时变漂移项进行拓展.电力现货价格的对数可定义为：

$$dX_t = \kappa(\alpha(t) - X_t)dt + \sigma dW_t + J_t dP_t(\omega) \tag{7.77}$$

对于 $\alpha(t)$，基于如下形式，模型把在峰顶的效应和非峰顶的效应合并进了价格过程：

$$\alpha(t) = \alpha_1 \text{peak}_t + \alpha_2 \text{offpeak}_t \tag{7.78}$$

其中，

$$\text{peak}_t = \begin{cases} 1, & \text{如果在峰顶周期内} \\ 0, & \text{其他} \end{cases} \tag{7.79}$$

$$\text{offpeak}_t = \begin{cases} 1, & \text{如果在非峰顶周期内} \\ 0, & \text{其他} \end{cases}$$

这同样是一个仿射过程. 基于同一种方法，转换条件特征函数被给定为：

$$\varphi(s,\theta,X_T \mid X_t) = E[\exp(isX_T) \mid X_t] = \exp(A(s,t,T,\theta) + B(s,t,T,\theta)X_t)$$

其中，$A(\cdot)$ 和 $B(\cdot)$ 满足如下复值常微分方程（ODE）系统：

$$\frac{\partial A(s,t,T,\theta)}{\partial t} = -\kappa\alpha B(s,t,T,\theta) - \frac{1}{2}\sigma^2(s,t,T,\theta) - \omega(\varphi(B(s,t,T,\theta)) - 1) \tag{7.80}$$

$$\frac{\partial B(s,t,T,\theta)}{\partial t} = \kappa B(s,t,T,\theta)$$

其边界条件为：

$$A(s,T,T,\theta) = 0, \qquad B(s,T,T,\theta) = is$$

对 $A(\cdot)$ 和 $B(\cdot)$ 求解式子（7.80），并应用相应的边界条件得：

$$\begin{aligned} A(s,t,T,\theta) = {} & i\alpha s(1 - \mathrm{e}^{-\kappa(T-t)}) - \frac{\sigma^2 s^2}{4\kappa}(1 - \mathrm{e}^{-2\kappa(T-t)}) \\ & + \frac{i\omega(1-2\psi)}{\kappa}(\arctan(\gamma s e^{-\kappa(T-t)}) - \arctan(\gamma s)) \\ & + \frac{\overline{\omega}}{2\kappa}\ln\left(\frac{1+\gamma^2 s^2 \mathrm{e}^{-2\kappa(T-t)}}{1+\gamma^2 s^2}\right) \end{aligned} \tag{7.81}$$

$$B(s,t,T,\theta) = is\mathrm{e}^{-\kappa(T-t)}$$

这里，令 $t=t_0<t_1<\cdots<t_N=T$；然后可得

$$\begin{aligned} L(s,t,T,\theta) &= \int_t^T \kappa\alpha(t) is\mathrm{e}^{-\kappa(T-t)}\,\mathrm{d}t \\ &= is\sum_{j=1}^{N} \alpha \mathrm{e}^{-\kappa(T-t_j)}(1 - \mathrm{e}^{-\kappa(t_j - t_{j-1})}) \end{aligned} \tag{7.82}$$

其中，

$$\alpha = \begin{cases} \alpha_1, & \text{如果}[t_{j-1},t_j]\text{在峰顶周期内} \\ \alpha_2, & \text{其他} \end{cases} \tag{7.83}$$

7.10.4 模型 2b：模型 1b 的时变版本

考虑将时变拓展到模型 1b(7.77)，其中，

$$\mathrm{d}X_t = \kappa(\alpha(t) - X_t)\mathrm{d}t + \sigma \mathrm{d}W_t + J_t^u \mathrm{d}P_t^u(\omega) + J_t^d \mathrm{d}P_t^d(\omega) \tag{7.84}$$

这里 $\alpha(t)$ 由式子(7.78) 定义.

类似于前面的模型，转换条件特征函数形式如下：

$$\begin{aligned} \varphi(s,\theta,X_T \mid X_t) &= E[\exp(isX_T) \mid X_t] \\ &= \exp(A(s,t,T,\theta) + B(s,t,T,\theta)X_t) \end{aligned}$$

其中，

$$
\begin{aligned}
A(s,t,T,\theta) &= i\alpha s(1-\mathrm{e}^{-\kappa(T-t)}) - \frac{\sigma^2 s^2}{4\kappa}(1-\mathrm{e}^{-2\kappa(T-t)}) \\
&\quad + \frac{\omega_u}{\kappa}\ln\left(\frac{1-is\gamma_u \mathrm{e}^{-\kappa(T-t)}}{1-is\gamma_u}\right) \\
&\quad + \frac{\omega_d}{\kappa}\ln\left(\frac{1-is\gamma_u \mathrm{e}^{-\kappa(T-t)}}{1-is\gamma_d}\right) \qquad (7.85) \\
B(s,t,T,\theta) &= is\mathrm{e}^{-\kappa(T-t)}
\end{aligned}
$$

这里 $L(s,\ t,\ T,\ \theta)$ 由式子 (7.82) 定义.

7.11 随机波动率定价模型

仅有跳跃不足以复制出现在电力价格中的峰度水平. Kaminski(1997) 和 Deng(1999) 强调在电力现货价格建模时加入随机波动率. 电力价格波动率随着时间而变化，它本身也是均值回复的 (参见 Goto 和 Karolyi[2003]). 为了捕捉随机波动率，各种双因素随机波动率模型由 Deng(1999)、Xiong(2004) 和 Villaplana(2002) 提出. 我们考虑用双因素仿射过程对电力现货价格建模.

模型 3a：随机波动率双因素跳跃扩散仿射过程

令 X_t 为电力现货价格的对数，V_t 为价格过程的波动率，它随着时间随机地演化.

$$
d\begin{bmatrix} X_t \\ V_t \end{bmatrix} = \begin{bmatrix} \kappa(\alpha - X_t) \\ \kappa_v(\alpha_v - V_t) \end{bmatrix} \mathrm{d}t + \begin{bmatrix} \sqrt{(1-\rho^2)V_t} & \rho\sqrt{V_t} \\ 0 & \sigma_v\sqrt{V_t} \end{bmatrix} \begin{bmatrix} \mathrm{d}W_t \\ \mathrm{d}W_v \end{bmatrix} + \begin{bmatrix} J_t \mathrm{d}P_t(\omega) \\ 0 \end{bmatrix} \qquad (7.86)
$$

其中，κ 是均值回复速率，α 是对数价格的长期均值，P_t 是到达速率 ω 的离散的一维标准泊松过程，振幅 J_t 是均值为 γ 的指数分布，J_t 的符号是参数 ψ 的伯努利随机变量分布. 两个随机变量 W_t 和 W_v 是相关系数为 ρ 的两个标准布朗运动. 同样，k_v 是波动率 V_t 的均值回复速率，α_v 是 V_t 的长期均值，σ_v 是 V_t 的波动率.

假定现货价格对数跳跃项由模型 1a 和模型 2a 定义. 接着，"跳跃转换" 给定如下：

$$
\begin{aligned}
\varphi\left(\begin{bmatrix} A(\cdot) \\ B(\cdot) \end{bmatrix}\right) &= \int_0^\infty (\psi\exp(A(\cdot)z) + (1-\psi)\exp(-A(\cdot)z)\ \frac{1}{\gamma}\exp\left(-\frac{z}{\gamma}\right)\mathrm{d}z \\
&= \frac{\psi}{1-A(\cdot)\gamma} + \frac{1-\psi}{1+A(\cdot)\gamma} \qquad (7.87)
\end{aligned}
$$

条件特征函数的形式是：

$$
\begin{aligned}
\varphi(s_x,s_v,\theta,X_T,V_T \mid X_t,V_t) &= E^\theta[\exp(is_x X_T + is_v V_T) \mid X_t,V_t] \\
&= \exp(A(s_x,s_v,t,T,\theta)X_t + B(s_x,s_v,t,T,\theta)V_t + C(s_x,s_v,t,T,\theta))
\end{aligned}
$$

其中，$A(\cdot)$-$B(\cdot)$ 和 $C(\cdot)$ 满足如下复值 Riccati 方程：

$$
\begin{cases}
\dfrac{\partial A(\cdot)}{\partial t} = \kappa A(\cdot) \\
\dfrac{\partial B(\cdot)}{\partial t} = \kappa_v B(\cdot) - \dfrac{1}{2}A(\cdot)(A(\cdot) + \rho\sigma_v B(\cdot)) \\
\qquad\qquad - \dfrac{1}{2}B(\cdot)(A(\cdot)\rho\sigma_v + B(\cdot)\sigma_v^2) \\
\dfrac{\partial C(\cdot)}{\partial t} = -\kappa\alpha A(\cdot) - \kappa_v\alpha_v B(\cdot) - \omega\left(\varphi\left(\begin{bmatrix} A(\cdot) \\ B(\cdot) \end{bmatrix}\right) - 1\right)
\end{cases} \qquad (7.88)
$$

边界条件为：

$$A(s_x,s_v,T,T,\theta)=is_x,\qquad B(s_x,s_v,T,T,\theta)=is_v,\qquad C(s_x,s_v,T,T,\theta)=0$$

对 $A(\cdot)$ 的第一个方程求解，并应用初始条件可以获得：

$$A(\cdot)=is_x\mathrm{e}^{-\kappa(T-t)}$$

然而，因为对于 $B(\cdot)$ 和 $C(\cdot)$ 并不存在闭型解，所以我们需要用数值方法求解.

7.12 模型参数估计

根据 Xiong(2004) 的研究，[39]我们讨论了怎样估计模型的参数. 仿射过程的弹性足以捕捉电力价格的特殊特征，如均值回复、季节性和“尖峰”. 此外，在特殊规则条件下，我们可以从离散样本观测的条件特征函数来探究信息，开发出易于计算的仿射过程参数的渐近有效估计. 此外，条件特征函数是唯一的，与通过傅里叶变换的条件密度函数包含同样的信息. 通过傅里叶变换，我们可以使用它来复原条件密度函数，实现最大似然估计. 这是 ML-CCF 估计的方法. 如果 N 维状态变量全部可观测，ML-CCF 估计就可以实现，因而可以获得的 ML-CCF 估计量是线性有效的（参见 Singleton[2001]).

然而，因为我们需要反复和精确地计算多变量傅里叶逆函数以使似然函数最大化，所以这个估计在高维（$N\geqslant 2$）情况下代价较高. 根据 Singleton(2001)，使用有限信息 ML-CCF (LML-CCF) 估计可以节省大量计算（参见 Singleton[2001]). 假定 $\{\boldsymbol{X}_t,\ t=1,\ 2,\ \cdots\}$ 是联合 CCF $\phi(s,\ \theta,\ \boldsymbol{X}_{t+1}\,|\,\boldsymbol{X}_t)$ 的 N 维状态变量的离散样本观测. 令 $\boldsymbol{\eta}_j$ 表示 N 维选择向量，其中第 j 个输入是 1，其他是 0. 定义 $X^j_{t+1}=\eta_j\cdot\boldsymbol{X}_{t+1}$；$\boldsymbol{X}_t$ 的条件密度函数 X^j_{t+1} 是具有标度 ξ 的 $\phi(\xi\eta_j,\ \theta,\ X_{t+1}\,|\,X_t)$ 的逆傅里叶变换：

$$f_j(X^j_{t+1},\theta|\boldsymbol{X}_t)=\frac{1}{2\pi}\int_{\mathcal{R}}\phi(\xi\eta_j,\theta,\boldsymbol{X}_{t+1}\,|\,\boldsymbol{X}_t)\mathrm{e}^{-i\xi\eta_j'X_{t+1}}\,\mathrm{d}s \tag{7.89}$$

这背后的基本思想是使用 $f_j(X^j_{t+1},\ \theta|\boldsymbol{X}_t)$ 中的信息而不是联合条件密度函数的信息：

$$f(X_{t+1},\theta|\boldsymbol{X}_t)=\frac{1}{(2\pi)^N}\int_{\mathcal{R}^N}\phi(s,\theta,X_{t+1}\,|\,X_t)\mathrm{e}^{-is'X_{t+1}}\,\mathrm{d}s \tag{7.90}$$

因此，估计涉及最多 N 个一维积分，而不是 N 维积分. 得到的估计量称为 LML-CCF 估计量. 尽管 LML-CCF 估计量没有采用任何与联合分布密度函数相关的信息，但是对于仿射扩散，它们一般比拟极大似然（QML）估计量更有效（参见 Singleton [2001]).

但对于那些不可见状态变量的多因素模型（如随机波动率模型），ML-CCF 或者 LML-CCF 估计量是不可获得的. 然而，一些文章讨论了基于 CCF 随机波动率模型估计量的方法. Singleton(1999) 提出了一个矩估计量的模拟方法（SMM－CCF)；Jiang 和 Knight(1999) 开发了矩估计量的矩系统（MSM)；Chacko 和 Viceira(2001) 考虑了所谓的矩的谱一般化方法 (SGMM). SGMM 相对于其他方法更易于计算（参见 Singleton[2001]). 为了处理随机波动率模型，Chacko 和 Viceira(2001) 从波动率的 CCF 得出了静态（非条件）特征函数[40]，并运用这个 CCF 获得了所谓的边际 CCF. 基于所谓边际 CCF(ML-MCCF) 来预测随机波动率模型的 ML 型估计被应用. 此外，基于边际 CCF 来预测随机波动率模型的 SGMM 估计量也被引入.

7.12.1 ML-CCF 估计量

如果概率密度有解析形式，ML 估计是随机过程参数估计的最常用方法. 它提供了一致性的

方法来解决参数估计问题，并且当样本大小增加时，ML 估计量成为最小方差无偏估计量. 假定 X 是概率密度函数 $f(X, \theta)$ 的 N 维连续随机变量，其中 $\theta=\{\theta_1, \cdots, \theta_k\}$ 是 k 个需要估计的未知不变参数. 给定在 $t=1, 2, \cdots, n$ 的观测序列 $\{\boldsymbol{X}_t\}$，在样本中对数似然函数给定如下：

$$L(\boldsymbol{X}_1, \cdots, \boldsymbol{X}_n, \theta) = \sum_{t=1}^{n} \ln f(\boldsymbol{X}_t, \theta) \tag{7.91}$$

基于 θ 估计量的最大似然可以通过最大化 $L(\cdot)$ 获得：

$$\hat{\theta}_{ml} = \arg\max_{\theta} L(\boldsymbol{X}_1, \cdots, \boldsymbol{X}_n, \theta) = \arg\max_{\theta} \sum_{t=1}^{n} \ln(f(\boldsymbol{X}_t, \theta)) \tag{7.92}$$

对于模型，我们采用样本的条件特征函数 $\phi(s, \theta, X_{t+1} | X_t)$ 是已知的，作为 $\boldsymbol{X}_t$ 仿射函数的指数是闭型的. 因此，给定 $\boldsymbol{X}_t$ 的 $\boldsymbol{X}_{t+1}$ 的条件密度函数可以通过条件特征函数的傅里叶变换获得：

$$f(X_{t+1}, \theta | X_t) = \frac{1}{(2\pi)^N} \int_{\mathcal{R}^N} \phi(s, \theta, X_{t+1} | X_t) \mathrm{e}^{-is' X_{t+1}} \mathrm{d}s \tag{7.93}$$

可以使用基于这个条件密度函数的标准 ML 估计获得样本的 ML-CCF 估计量：

$$\hat{\theta}_{CCF} = \arg\max_{\theta} \sum_{t=1}^{n} \ln(f(\boldsymbol{X}_{t+1}, \theta | \boldsymbol{X}_t)) \tag{7.94}$$

以模型 1a(7.65) 为例. 给定 $\boldsymbol{X}_t$ 属于样本中，$\boldsymbol{X}_{t+1}$ 的条件密度函数形式如下：

$$\begin{aligned} f(X_{t+1}, \theta | \boldsymbol{X}_t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi(s, \theta, X_{t+1} | X_t) \mathrm{e}^{-isX_{t+1}} \mathrm{d}s \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathrm{e}^{-isY_t} h(\theta, s) \mathrm{d}s \end{aligned} \tag{7.95}$$

其中，

$$Y_t = (X_{t+1} - \alpha) - \mathrm{e}^{-\kappa}(X_t - \alpha), \tag{7.96}$$

和

$$\begin{aligned} h(\theta, s) = \exp\Big(&-\frac{\sigma^2 s^2}{4\kappa}(1 - \mathrm{e}^{-2\kappa}) + \frac{i\omega(1 - 2\psi)}{\kappa}(\arctan(\gamma s \mathrm{e}^{-\kappa}) - \arctan(\lambda s)) \\ &+ \frac{\omega}{2\kappa} \ln\Big(\frac{1 + \gamma^2 s^2 \mathrm{e}^{-2\kappa}}{1 + \gamma^2 s^2}\Big) \end{aligned} \tag{7.97}$$

为了计算这个积分 (7.95)，我们定义

$$F(Y_t, \theta) = f(X_{t+1}, \theta | X_t) = \frac{1}{2\pi} \lim_{R \to \infty} \int_{-R}^{R} \mathrm{e}^{-isY_t} h(\theta, s) \mathrm{d}s \tag{7.98}$$

值得注意的是，在 s 中 $|h(\theta, s)|$ 是连续的，且 $|h(\theta, s)| \leqslant -\frac{\sigma^2 s^2}{4k}(1 - \mathrm{e}^{-2k})$. 因此，我们可以把整个积分截断成有限区间 $[-R, R]$ 上的积分，在这个区间之外，被积函数 $h(\theta, s)$ 非常小，可以忽略. 接着，对于 R 的这个选择，

$$F(Y_t, \theta) \approx \frac{1}{2\pi} \int_{-R}^{R} \mathrm{e}^{-isY_t} h(\theta, s) \mathrm{d}s \tag{7.99}$$

同样，将 Y_t 离散化到 M 个子区间，以使得：[41]

$$Y_n = n\Delta Y_t = n\Big(\frac{Y_t}{M}\Big)$$

$$s_\kappa = \kappa\Delta s = \kappa\left(\frac{R}{M}\right)$$

$$F(Y_t,\theta) \approx \frac{1}{2\pi}\frac{R}{M}\sum_{n=-M}^{M-1'}\left(\mathrm{e}^{-in\kappa\frac{RY_t}{M}}h\left(\theta,\frac{nR}{M}\right)\right) \tag{7.100}$$

如果安排$\frac{RY_t}{M}=2\pi$，那么可得：

$$F(Y_t,\theta) \approx \frac{1}{2\pi}\frac{R}{M}\sum_{n=-M}^{M-1'}\left(\mathrm{e}^{-in\kappa\frac{2\pi}{M}}h\left(\theta,\frac{nR}{M}\right)\right) \tag{7.101}$$

通过$h\left(\theta,\ \frac{nR}{M}\right)$的离散傅里叶变换（DFT）可以近似$F(Y_t,\ \theta)$，式子（7.95）的积分可以在$s$值的合适划分上通过快速傅里叶变换（FFT）算法估计出来.

7.12.2 ML-MCCF 估计量

如果所有状态变量是可观测的，ML-MCCF 估计量是渐近有效的. 但是对于那些没有可观测状态变量的多因素模型，如模型 3a 和模型 3b，ML-CCF 估计量不能直接获得. 如果期权价格可得，隐含波动率能够从市场上观测到的期权价格计算得出. 从期权价格来估计隐含波动率的各种数值方法已被提出（参见 Dupire[1994]、Coleman、Li 和 Verma[1999] 及 Hamida 和 Cont[2004]）. 然后可以使用那些数值作为波动率数据，实现 ML-CCF 估计.

但是在我们的案例中，期权价格是不可获得的. 根据 Chacko 和 Viceira(2001)，我们可以对来自对数价格和波动率的联合 CCF 的不可观测变量（波动率）积分，并令$s_v=0$，但是同样需要运用波动率信息（在我们的例子中不可行）. Singleton 的 SMM 方法通过模拟积分消去了在 CCF 中的不可观测变量. 这需要大量波动率的模拟路径，并会耗费大量时间. 而且，由于模拟中的离散化，这会导致估计的偏差（参见 Chacko 和 Viceira[2001]）. 同时，与将要引入的 SGMM 方法相比，ML-MCCF 估计避免了所谓的特别矩条件选择问题，更容易实现随机波动模型.

以模型 3a 为例. 波动率遵循单位根过程，如下所示：

$$\mathrm{d}V_t = \kappa_v(\alpha_v - V_t)\mathrm{d}t + \sigma_v\sqrt{V_t}\mathrm{d}W_v \tag{7.102}$$

单位根过程的无穷小量产生器是：

$$Lf(v) = \frac{\sigma_v^2 v}{2}\frac{\partial^2 f}{\partial v^2} + \kappa_v(\alpha_v - v)\frac{\partial f}{\partial v} \tag{7.103}$$

令μ_t是分布函数，然后它向前对 Kolmogorov 方程（7.103）求解：

$$\mu_t(Lf) = \frac{\mathrm{d}}{\mathrm{d}t}\mu_t(f) \tag{7.104}$$

其中，$\mu_t(f)=\int f(v)\mathrm{d}\mu_t$.

特别地，令μ是波动率的静态特征函数. 在这种情形中，当$f(v)=\mathrm{e}^{iuv}$和$\hat{\mu}(u)=\mu(\mathrm{e}^{iuv})$时，有：

$$\begin{aligned} L\mathrm{e}^{iuv} &= -\frac{\sigma_v^2 v}{2}u^2\mathrm{e}^{iuv} + i\kappa_v(\alpha_v - v)u\mathrm{e}^{iuv} \\ &= \left(iv\left(\frac{i\sigma_v^2 u^2}{2} - \kappa_v u\right) + i\kappa_v\alpha_v u\right)\mathrm{e}^{iuv} \end{aligned} \tag{7.105}$$

和

$$\mu(Le^{iuv}) = \left(\frac{i\sigma_v^2 u^2}{2} - \kappa_v u\right)\int iv e^{ivu} du + i\kappa_v \alpha_v u \int e^{iuv} du$$
$$= \left(\frac{i\sigma_v^2 u^2}{2} - \kappa_v u\right)\frac{d\hat{\mu}(u)}{du} + i\kappa_v \alpha_v u\, \hat{\mu}(u) \tag{7.106}$$

因为$\frac{d\mu(\cdot)}{dt}=0$，所以有：

$$\left(\frac{i\sigma_v^2 u^2}{2} - \kappa_v u\right)\frac{d\hat{\mu}(u)}{du} + i\kappa_v \alpha_v\, \hat{\mu}(u) = 0 \tag{7.107}$$

其中$\hat{\mu}(0)=1$. 随后式(7.107) 的解有如下形式：

$$\hat{\mu} = \left(1 - \frac{iu\sigma_v^2}{s\kappa_v}\right)^{\frac{-2\kappa_v\alpha_v}{\sigma_v^2}} \tag{7.108}$$

回顾，这个模型中的对数价格和波动率的联合条件特征函数定义如下：

$$\phi(s_x, s_v, \theta, X_T, V_T | X_t, V_t) = \exp(A(s_x, s_v, t, T, \theta)X_t + B(s_x, s_v, t, T, \theta)V_t + C(s_x, s_v, t, T, \theta))$$

其中 $A(\cdot)$、$B(\cdot)$ 和 $C(\cdot)$ 是系统 (7.88) 的解. 当随机波动率 V_t 不可观测时，我们无法直接从对数价格和波动率的联合条件特征函数估计随机模型的参数. 定义边际条件特征函数为：

$$\phi(s_x, \theta, X_T | X_t) = \int_0^\infty \phi(s_x, 0, \theta, X_T, V_T | X_t, V_t) du$$
$$= e^{A(s_x,0,t,T,\theta)X_t + C(s_x,0,t,T,\theta)} \int_0^\infty e^{B(s_x,0,t,T,\theta)V_t} du$$
$$= e^{A(s_x,0,t,T,\theta)X_t + C(s_x,0,t,T,\theta)}\, \hat{\mu}(-iB(s_x,0,t,T,\theta)) \tag{7.109}$$

运用公式 (7.108)，可以获得边际条件特征函数：

$$\phi(s_x, \theta, X_T | X_t) = e^{A(s_x,0,t,T,\theta)X_t + C(s_x,0,t,T,\theta)} \cdot \left(1 - \frac{B(s_x,0,t,T,\theta)\sigma_v^2}{2\kappa_v}\right)^{\frac{-2\kappa_v\alpha_v}{\sigma_v^2}} \tag{7.110}$$

通过傅里叶变换，边际条件密度函数给定为：

$$f(X_{t+1}, \theta | X_t) = \frac{1}{2\pi}\int_{\mathcal{R}} \phi(s_x, \theta, X_{t+1} | X_t) e^{-is_x X_{t+1}} ds \tag{7.111}$$

然后，给定样本 $\{X_t,\ t=1,\ \cdots,\ n\}$，基于观测变量（电力价格）的边际分布，可以实现最大似然估计，得到 ML-MCCF 估计量为：

$$\theta_{MCCF} = \arg\max_\theta \sum_{t=1}^{n-1} \ln f(X_{t+1}, \theta | X_t) \tag{7.112}$$

值得注意的是，因为我们仅仅依赖前面时段电力价格的水平，所以失去了效率. 正如 Chacko 和 Viceira(2001) 所指出，点估计（包括即将讨论的 SGMM 估计量）是有偏和不一致的. 另外，偏差的理论值难以计算，因为我们没有 $B(\cdot)$ 和 $C(\cdot)$ 的闭型解. 根据 Chacko 和 Viceira(2001)，我们试图通过自举方法修正这个偏差. 特别是，当给定参数 θ_0 时，我们模拟了 500 条路径. 对于每一条路径，存在 19 704 个小时观测（跟实际数据同样长）. 从模拟路径获得的参数$\hat{\theta}_i$$(i=1,\ \cdots,\ n)$ 对于每一个参数给出了一个分布. 我们将认为那些参数之间的均值和给定参数之间的差为偏差，即

$$bias = \theta_0 - \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i \tag{7.113}$$

在我们的设置中，$n=500$.[42]

7.12.3 光谱广义矩估计量

Chacko 和 Viceira(2001) 构建了光谱广义矩估计量. 这种方法在复变环境中是必要的广义矩方法. 广义矩估计量是统计学和计量经济学中最基本的估计方法之一（参见 Hansen[1982]). 与需要模型全部情况和概率分布的极大似然估计不一样，广义矩估计并不需要全面了解信息和较强的分布假设. GMM 估计量最适合用于研究那些仅部分规定的模型，它们是似然类型估计量的合适替代物.

定义 1 假定有随机变量的集合 $\{x_t,\ t=1,\ 2,\ \cdots\}$. 令 $\boldsymbol{\theta}=\theta_1,\ \cdots,\ \theta_k$ 是将要估计的真实值 $\boldsymbol{\theta}_0$ 的未知元组；θ_0，θ 属于参数空间 Θ. q 维函数向量 $\boldsymbol{m}(x_t,\ \theta)$ 称为 **(无条件) 矩函数**，如果以下矩条件满足：

$$E[\boldsymbol{m}(x_t,\boldsymbol{\theta}_0)] = 0 \tag{7.114}$$

值得注意的是，θ 是 k 元向量，$E[\boldsymbol{m}(x_t,\ \boldsymbol{\theta}_0)]=0$ 包含 q 个方程. 如果有同将要估计的参数一样多的矩条件 ($q=k$)，那么可以轻松地解决 k 个方程中的 k 个未知变量来获得估计. 如果拥有的矩条件比未知量要少 ($q<k$)，我们就可以确定 $\boldsymbol{\theta}$. 在这个案例中，我们能通过所谓的权重函数“增加”更多的矩条件（在广义矩方法中通常被称为“工具”；参见 Matyas[1999]). 如果有比未知量更多的函数 ($q>k$)，就会产生一个过度确定问题. 这些过度确定的案例非常容易产生，矩估计量并没有很好的定义. 矩条件的不同选择会导致不同的估计. 广义矩是解决这类过度问题的方法.

以标准线性回归问题为例：

$$y = x'\boldsymbol{\theta}_0 + \varepsilon \tag{7.115}$$

这里，y 是响应变量，$\boldsymbol{x}=[x_1,\ x_2,\ \cdots,\ x_k]'$ 是回归量的 k 维向量，$\boldsymbol{x}'$ 是它的转置，$\boldsymbol{\theta}=[\theta_1,\ \cdots,\ \theta_k]'$ 是具有真实值 $\boldsymbol{\theta}_0$ 的参数的未知向量. 假定 ε 的期望为 0，且与 x 不相关. 使用迭代期望法则，我们发现：

$$E[x\varepsilon] = E[E[x\varepsilon\,|\,x]] = E[xE[\varepsilon\,|\,x]] = 0 \tag{7.116}$$

因此，我们有矩函数 $\boldsymbol{m}((x,\ y),\ \boldsymbol{\theta})=\boldsymbol{x}(y-\boldsymbol{x}'\theta)$. 通过假设

$$E[\boldsymbol{m}((\boldsymbol{x},y),\boldsymbol{\theta}_0)] = E[\boldsymbol{x}(y-\boldsymbol{x}'\boldsymbol{\theta}_0)] = E[\boldsymbol{x}\varepsilon] = 0. \tag{7.117}$$

这些矩函数被较好地定义了.

假定响应变量的 $n>k$ 个观测是可获得的，如 $y_1,\ y_2,\ \cdots,\ y_n$. 随同每个观测响应 y_t 一起，我们有回归量 $\boldsymbol{x}_t$ 的 k 维观测向量. 因为 $\boldsymbol{x}_t$ 是一个 k 维向量，我们有与要估计的参数一样多的矩条件. 如果假定强大数法则成立，则几乎必然有：

$$\frac{1}{n}\sum_{t=1}^{n} m((x,y),\hat{\boldsymbol{\theta}}_0) \rightarrow E[\boldsymbol{m}((x,y),\boldsymbol{\theta}_0)] = 0$$

所以对于这个模型来说，矩方法 (MM) 估计量仅仅是下式的解：

$$\frac{1}{n}\sum_{t=1}^{n}x_t(y_t - x_t{}'\hat{\boldsymbol{\theta}}_n) = 0 \tag{7.118}$$

它给出

$$\hat{\boldsymbol{\theta}}_n = \left(\sum_{t=1}^{n}\boldsymbol{x}_t\boldsymbol{x}_t'\right)^{-1}\sum_{t=1}^{n}\boldsymbol{x}_t y = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'y \tag{7.119}$$

其中，$\boldsymbol{X}=[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$，$\boldsymbol{y}=[y_1, \cdots, y_n]'$. 因此，普通最小二乘法估计量是矩方法估计量.

值得注意的是，关于误差项 ε 我们列出了相对少的信息. 对于最大似然估计来说，我们需要给定误差项 ε 的分布，还需要给定自相关性和异方差性，而这些在形成矩条件时也不需要.

现在放弃一些观测变量的误差项期望为 0 的假定，通过与一些观测“工具”不相关的误差项，我们能够直接详细说明矩条件. 再次考虑先前的模型. 这次，我们不假定误差项期望为 0，但它与回归量仍然是不相关的. 假定有 q 维观测工具 $\boldsymbol{z}(q>k)$ 和 $E[\boldsymbol{z}\varepsilon]=0$. 因此有矩条件：

$$E[\boldsymbol{z}\varepsilon] = E[\boldsymbol{z}(y - x'\boldsymbol{\theta}_0)] = 0 \tag{7.120}$$

和矩函数：

$$\boldsymbol{m}((x,y,z),\theta) = \boldsymbol{z}(y - x'\boldsymbol{\theta}) \tag{7.121}$$

如果 $q=k$，那么这同样是一个明确的问题. 令 $\boldsymbol{z}_t$ 表示相应于 y_t 的工具的 k 维观测向量. 假定强大数定律成立，所以几乎必然有：

$$\frac{1}{n}\sum_{t=1}^{n}\boldsymbol{m}((x_t,y,\boldsymbol{z}_t),\hat{\boldsymbol{\theta}}_0) \to E[\boldsymbol{m}((x,y,\boldsymbol{z}),\boldsymbol{\theta}_0)] = 0 \tag{7.122}$$

因此，求解下面方程：

$$\frac{1}{n}\sum_{t=1}^{n}\boldsymbol{z}(y - x'\hat{\boldsymbol{\theta}}_t) = 0 \tag{7.123}$$

可得：

$$\hat{\boldsymbol{\theta}}_n = \left(\sum_{t=1}^{n}\boldsymbol{z}_t\boldsymbol{x}_t{}'\right)^{-1}\sum_{t=1}^{n}\boldsymbol{z}_t y = (\boldsymbol{Z}'X)^{-1}\boldsymbol{Z}'y \tag{7.124}$$

其中，$\boldsymbol{Z}=[\boldsymbol{z}_1, \cdots, \boldsymbol{z}_n]$.

样本条件特征函数的定义蕴涵着：

$$E[\exp(is\cdot\boldsymbol{X}_T) - \phi(s,\theta,\boldsymbol{X}_T|\boldsymbol{X}_t)] = 0, s\in\mathcal{R}^n. \tag{7.125}$$

通过取这个函数的实部和虚部，可得到如下的配对矩条件：

$$\begin{aligned}E[\mathrm{Re}(\exp(is\cdot\boldsymbol{X}_T) - \phi(s,\theta,\boldsymbol{X}_T|\boldsymbol{X}_t))] &= 0\\ E[\mathrm{Im}(\exp(is\cdot\boldsymbol{X}_T) - \phi(s,\theta,\boldsymbol{X}_T|\boldsymbol{X}_t))] &= 0\end{aligned} \tag{7.126}$$

因此，可以定义矩函数集合：

$$\boldsymbol{m}(s,\theta,\boldsymbol{X}_T,\boldsymbol{X}_t) = \varepsilon_t(s,\boldsymbol{\theta},\boldsymbol{X}_T,\boldsymbol{X}_t) = \begin{bmatrix}\varepsilon_t^{\mathrm{Re}}(s,\theta,\boldsymbol{X}_T,\boldsymbol{X}_t)\\ \varepsilon_t^{\mathrm{Im}}(s,\theta,\boldsymbol{X}_T,\boldsymbol{X}_t)\end{bmatrix}$$

$$\varepsilon_t^{\mathrm{Re}}(s,\boldsymbol{\theta},\boldsymbol{X}_T,\boldsymbol{X}_t) = \mathrm{Re}(\varepsilon_t(s,\boldsymbol{\theta},\boldsymbol{X}_T,\boldsymbol{X}_t)) = \mathrm{Re}(\exp(is\cdot\boldsymbol{X}_T) - \phi(s,\boldsymbol{\theta},\boldsymbol{X}_T|\boldsymbol{X}_t))$$

$$\varepsilon_t^{\mathrm{Im}}(s,\boldsymbol{\theta},\boldsymbol{X}_T,\boldsymbol{X}_t) = \mathrm{Im}(\varepsilon_t(s,\boldsymbol{\theta},\boldsymbol{X}_T,\boldsymbol{X}_t)) = \mathrm{Im}(\exp(is\cdot\boldsymbol{X}_T) - \phi(s,\boldsymbol{\theta},\boldsymbol{X}_T|\boldsymbol{X}_t))$$

更一般地，可以通过加入“工具”或者“权重函数”的集合来获得约束. 基于条件特征函数的矩函数可定义为：[43]

$$m(s,\theta,X_T,X_t)=\varepsilon_t(s,\theta,X_T,X_t)\otimes p(X_t)$$

其中，$p(X_t)$ 是 $\varepsilon_t(s,\theta,X_T,X_t)$ 的“工具”，$\otimes$是 Kronecker 积. 光谱广义矩估计量形式为：

$$\hat{\theta}_{\text{SGMM}}=\arg\min_{\theta}\left[\frac{1}{n}\sum_{t=1}^{n}m(s,\theta,X_T,X_t)\right]'W_n\left[\frac{1}{n}\sum_{t=1}^{n}m(s,\theta,X_T,X_t)\right],\theta\in\Theta$$

正如对于其他广义矩估计量一样，当最优权重矩阵 $W_n=S^{-1}$时，光谱广义矩估计量的渐近方差是最小的，其中 S 是矩函数的协方差矩阵（参见 Chacko 和 Viceira [2001]). 在通常正则性条件下，根据 Chacko 和 Viceira(2001)，光谱广义矩估计量$\hat{\theta}_{\text{SGMM}}$继承了广义矩估计量的最优属性，如一致性和渐近正态性（参见 Hansen [1982]).

现在能够应用光谱广义矩方法来估计像模型 3a 和模型 3b 那样的随机波动率模型. 回想样本的边际条件特征函数给定为：

$$\phi(s_x,\theta,X_{t+1}\mid X_t)=e^{A(s_x,0,t,\theta)X_t+C(s_x,0,t,\theta)}\left(1-\frac{B(s_x,0,t,\theta)\sigma_v^2}{2\kappa_v}\right)^{-\frac{2\kappa_v\alpha_v}{\sigma_v^2}} \tag{7.127}$$

其中对于模型 3a 和模型 3b，$A(\cdot)$、$B(\cdot)$ 和 $C(\cdot)$ 分别是系统（7.88）的解. 给定样本 $\{X_t, t=1,\cdots,T\}$，有如下矩函数：

$$m(s_x,X_t,\theta)=\varepsilon_t(s_x,X_t,\theta)=\begin{bmatrix}\varepsilon_t^{\text{Re}}(s_x,X_t,\theta)\\ \varepsilon_t^{\text{Im}}(s_x,X_t,\theta)\end{bmatrix}$$

$$\varepsilon_t^{\text{Re}}(s_x,X_t,\theta)=(\cos(s_xX_{t+1})-\text{Re}(\phi(s_x,0,X_{t+1}\mid X_t)))\otimes p(X_t)$$

$$\varepsilon_t^{\text{Im}}(s_x,X_t,\theta)=(\sin(s_xX_{t+1})-\text{Im}(\phi(s_x,0,X_{t+1}\mid X_t)))\otimes p(X_t) \tag{7.128}$$

其中，$\phi(s_x, 0, X_{t+1}\mid X_t)$ 由公式（7.127）定义. 根据 Chacko 和 Viceira（2001），通过把 $s_x=n$ 简单代入公式（7.128）可以计算 n 阶条件矩.

为了构建矩函数，经过设定 $s_x=1, 2, \cdots, 6$ 和 $p(X_t)$ 为 1 的 T 维向量，Xiong (2004) 使用了前 6 个光谱矩. 估计是有偏的和不一致的（参见 Matyas[1999]). 使用这个方法，Xiong 发现了模型 3a 的参数估计，如表 7.5 所示.

表 7.5

	估 计	标准差	T 比率
κ	0.115 2	0.002 3	50.087 0
α	−0.508 6	0.041 3	−12.314 8
ω	1.081 4	0.017 0	63.611 8
ψ	−0.578 1	0.009 0	64.233 3
γ	0.352 2	0.006 5	54.184 6
ρ	−0.971 0	0.001 4	−693.571 4
$\kappa_{\bar{\omega}}$	0.548 3	0.031 5	17.406 3
$\alpha_{\bar{\omega}}$	0.026 1	0.002 0	13.050 0
$\sigma_{\bar{\omega}}$	0.152 9	0.017 1	8.941 5
对数似然	−11 345.801 3		

资料来源：Xiong(2004).

7.12.4 模拟

经过模型 1b 模拟路径的叠加，图 7.14 展示了尖峰电力价格. 图 7.14 还包括了尖峰电力价格的样本图，一个典型的模拟路径就是 95%分位数和 5%分位数的模拟.

经过模型 1b 模拟路径的叠加，图 7.15 展示了非峰值的电力价格. 图 7.15 同样包括了尖峰电力价格的样本图，一个典型的模拟路径就是 95%分位数和 5%分位数的模拟.

图 7.16 和图 7.17 显示了尖峰电力价格（模型 1b）和非峰值电力价格（模型 1b）各自模拟价格过程的比较. 值得注意的是，图 7.16 是去季节化的尖峰电力价格变化的直方图. 覆盖黑

线是相应的典型模拟路径的对数回报率分布. 注意, 这个模型低估了小变化的数目, 高估了中间大小的变化.

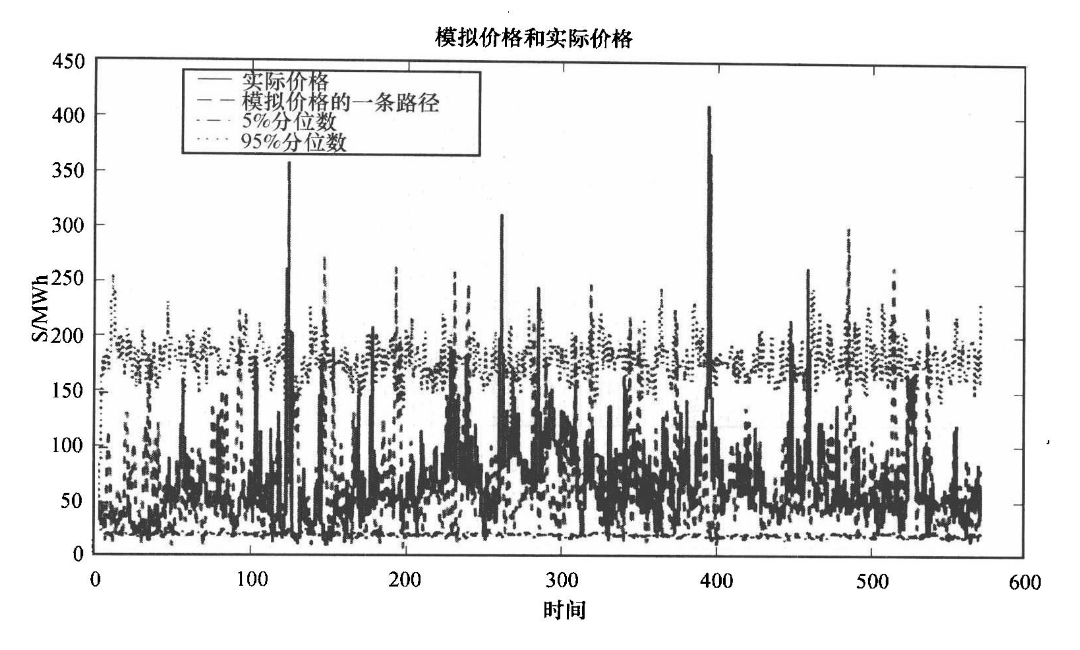


图 7.14
资料来源: Xiong(2004).

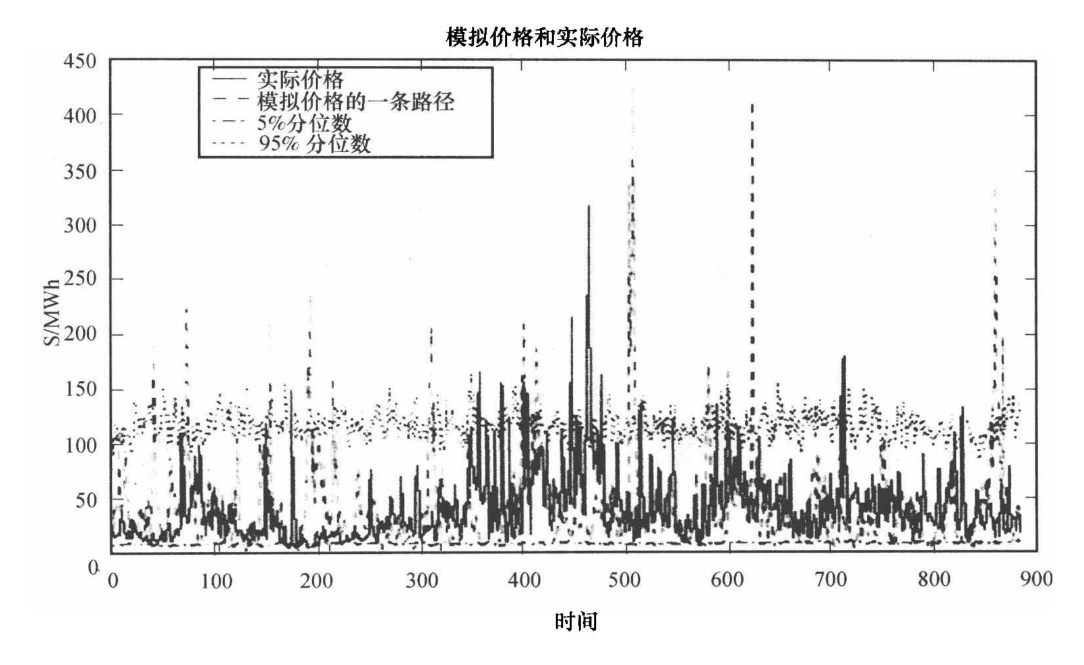


图 7.15
资料来源: Xiong(2004).

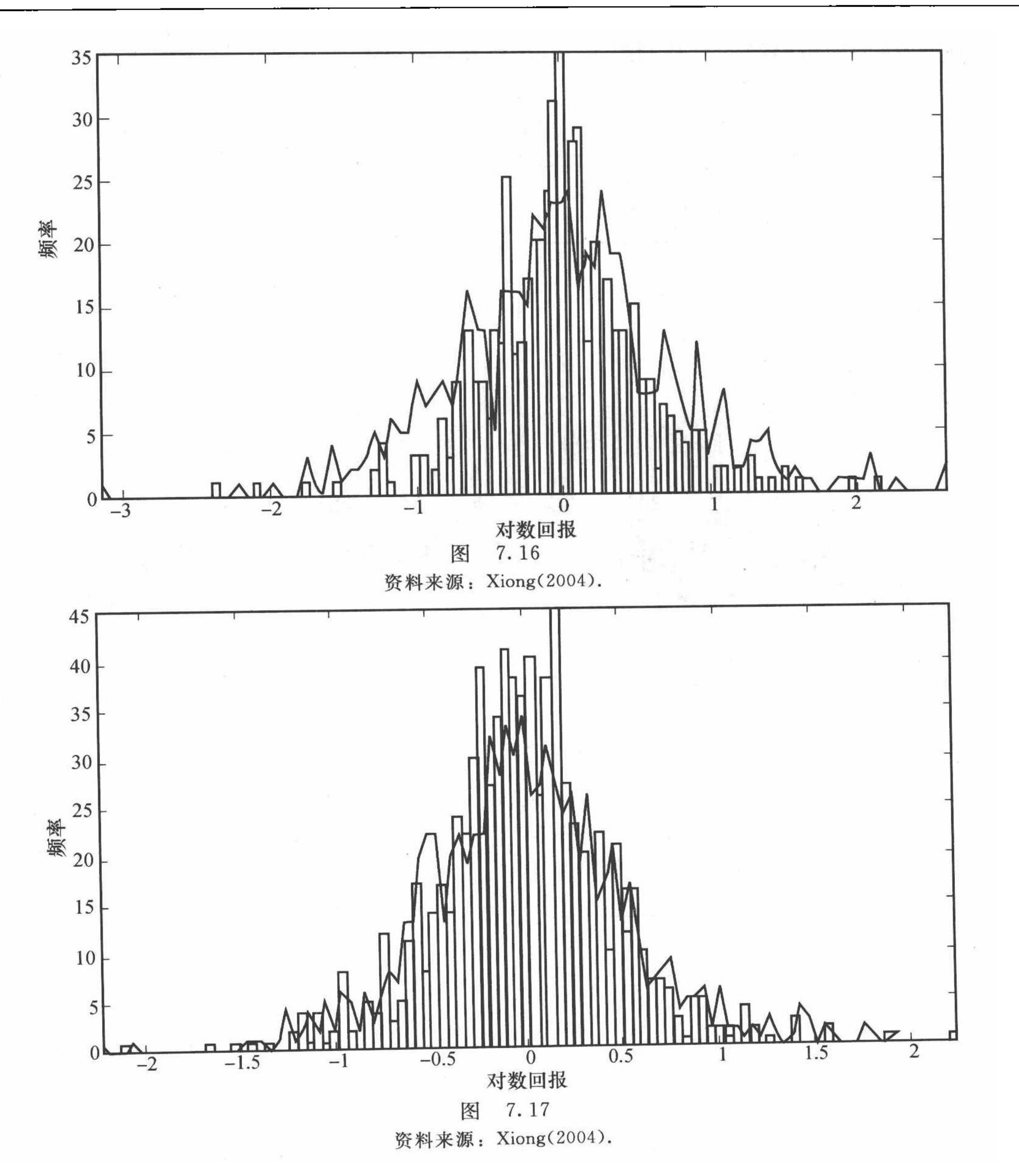


图 7.16

资料来源：Xiong(2004).

图 7.17

资料来源：Xiong(2004).

7.13 应用 Matlab 估计参数

由 Lei Xiong 和 Anthony Ware 编写的 Matlab 代码的全部清单可参阅附录 B，那些代码使用由前面 Xiong(2004) 给定的最大似然方法估计了模型参数. Xiong 和 Ware 用来标定与估计模型参数（存储于数组并用于函数）的市场数据来自 Alberta Nordic Pool（但未提供）.

7.14 能源商品模型

对于像裂解价差合约[44]这样的工具，用来对能源商品价格建模的普通均值回复模型是

Schwartz-Ross 模型：

$$dS_t = \alpha(L - \ln S_t) S_t dt + \sigma S_t dW_t \tag{7.129}$$

其中，W_t 是标准布朗运动，α，L 和 σ 全是正常数. 在这个模型中，S_t 均值回复到长期均值 $\hat{L} = e^L$. 对 $X_t = \ln S_t$ 应用 Ito 引理，得到：

$$dX_t = \alpha\left(L - \frac{\sigma^2}{2\alpha} - X_t\right) dt + \sigma dW_t \tag{7.130}$$

这是一个 Ornstein-Uhlenbeck 过程. 与回复到对数价格的长期均值不同，价格水平本身的回复能用 Dixit-Pinkyck 模型建模：

$$dS_t = \alpha(L - S_t) S_t dt + \sigma S_t dW_t \tag{7.131}$$

其中，W_t 是标准布朗运动，α，L 和 σ 全是正常数. 这些参数的每一个都是时变的和随机的.

与其他像糖和黄金这样类型商品价格相比，能源商品价格没有可辨别的趋势. 正如图 7.18 所示，原油现货价格（West Texas Intermediate at Cushing，Oklahoma）、燃料油（纽约港）、无铅汽油（纽约港）和天然气（Henry Hub，LA）是随机波动的.

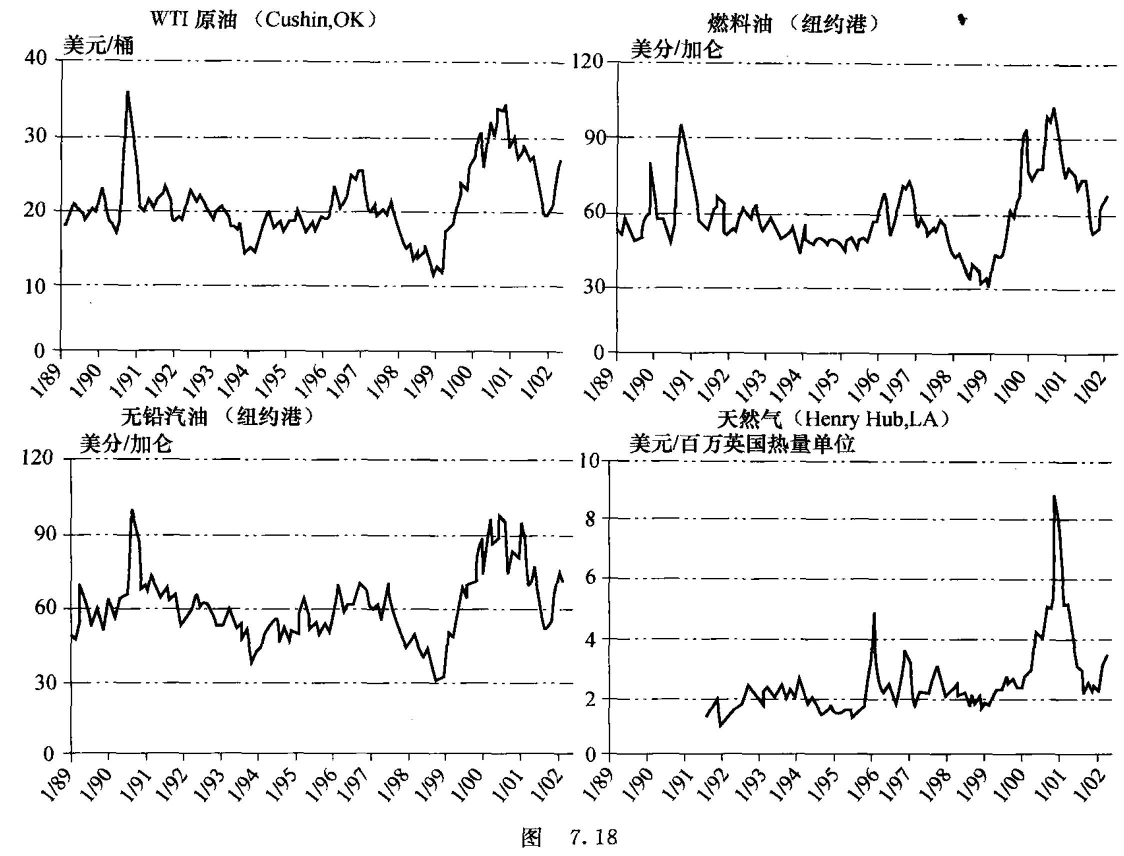


图 7.18

资料来源：Commodity Futures Trading Commission（参见 Energy Information Adminis-tration/Derivatives and Risk Managementin Energy Industries，U. S. Dept. of Energy，(2002)，pg. 10）.

燃料油和汽油价格常常与原油价格一起波动，而天然气现货市场（将在下章讨论）会毫无征兆地周期性到达尖峰. 通常，与其他类型商品相比能源商品，有着更高的波动率. 电力商品有

着最高的波动率，而金融商品波动率最低，正如图 7.19 所示，它展现了 1992 年至 2001 年各个商品的平均年度历史现货市场价格波动率.

商品	平均每年波动率（%）	市场	周期
电力			
California-Oregon Border	309.9	现货一峰值	1996—2001
Cinergy	435.7	现货一峰值	1996—2001
Palo Verde	304.5	现货一峰值	1996—2001
PJM	389.1	现货一峰值	1996—2001
天然气和石油			
轻质原油，LLS	38.3	现货	1989—2001
车用汽油，NYH	39.1	现货	1989—2001
燃料油，NYH	38.5	现货	1989—2001
天然气	78.0	现货	1992—2001
金融			
联邦资金利率	85.7	现货	1989—2001
股票指数，S&P500	15.1	现货	1989—2001
长期国库券，30 年	12.6	现货	1989—2001
金属			
铜，LME GradeA	32.3	现货	January 1989—August 2001
金条，Handy & Harman，NY	12.0	现货	1989—2001
银条，Handy & Harman，NY	20.2	现货	January 1989—August 2001
白金，Producers	22.6	现货	January 1989—August 2001
农产品			
咖啡，BH OM Arabic	37.3	现货	January 1989—August 2001
糖，World Spot	99.0	现货	January1989—August 2001
玉米，N. Illinois River	37.7	现货	1994—2001
大豆，N. Illinois River	23.8	现货	1994—2001
棉花，East TX & OK	76.2	现货	January 1989—August 2001
浓缩橙汁，Florida Citrus Mutual	20.3	现货	September1998-December 2001
肉类			
牛肉，Amarillo	13.3	现货	January 1989—August 2001
猪胸肉	71.8	现货	January 1989—August 2001

图 7.19

资料来源：Energy Information Administration/Derivatives and Risk Management in the Energy Industries，U. S. Dept. of Energy(2002).

我们使用这些波动率对模型（7.129）～（7.131）中的 σ 进行估计.

7.15 天然气

7.15.1 天然气市场

天然气是一种重要的能源，消费者经常需要和使用它作为像电力一样的其他能源的廉价替代物. 天然气[45]的主要用途是作为家庭能源和化学原料供热、发电. 天然气大约占北美总能源消费量的$\frac{1}{4}$. 北美天然气管制规则随着 1978 年美国天然气政策法案的颁布已经导致动态、高度竞争性的市场，这个市场价格剧烈波动. 在价格管制之前，对于国内原油和天然气衍生品市场是

有限的.在价格管制下，美国能源部（DOE）、联邦能源管制委员会（FERC）、州公用事业委员会（PUC）直接或间接地控制国内原油价格、石油产品、源头天然气、管道天然气和零售天然气服务.

解除管制环境的后果是，像天然气生产者这样的市场参与者发现他们暴露于价格波动以及交易方的履约风险，这使得风险管理的需求大幅增加，从而促进天然气即期与远期市场的发展.因此，纽约商品交易所（NYMEX）于1990年4月启动了世界上第一个天然气期货合约.自从那以后，在这个波动的市场，天然气期货合约被广泛用于对冲价格波动.标准化的期货合约、相对较小的合约大小、可替代性、实物交割需求的缺乏和严格的业绩要求吸引了众多市场参与者，如天然气生产者、商人、加工商、公用事业、终端用户和投机者.成交量和持仓量迅速增加，使得天然气合约成为纽约商品交易所历史上增长最快的品种.在2002年，纽约商品交易所市场的日平均成交量超过了97 000张合约，它是北美天然气的平均日度消费的几倍.图7.20显示了美国天然气的主要定价点（中心).路易斯安那州的亨利是一个主要的中点.

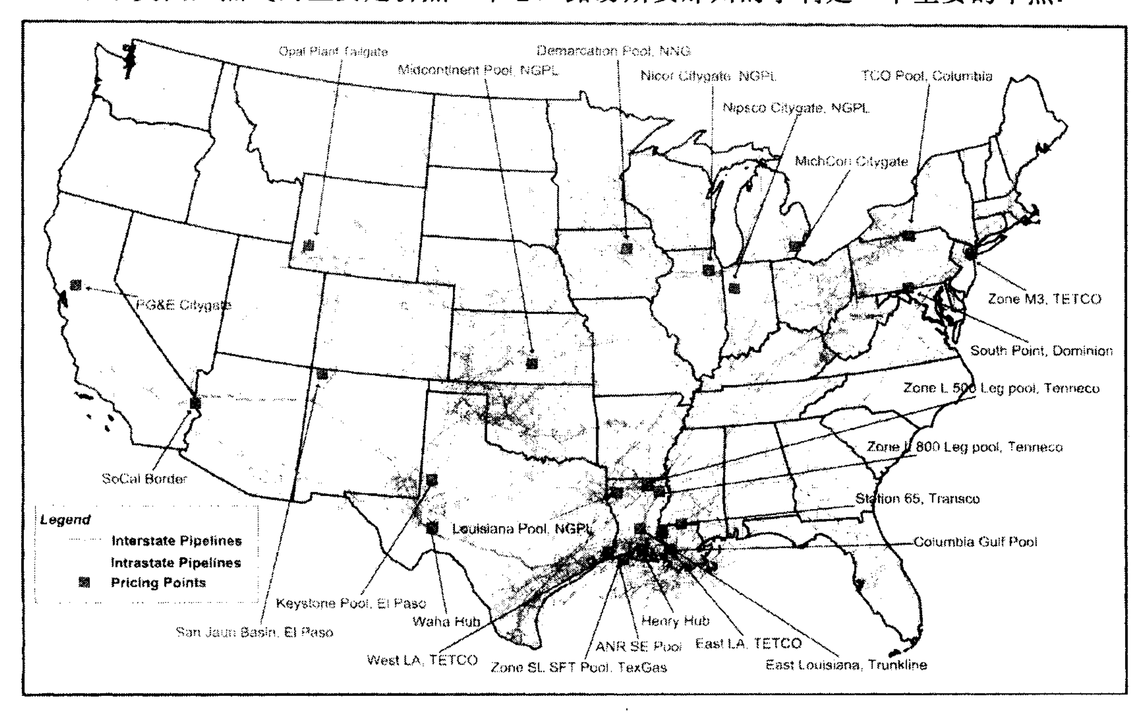


图 7.20

资料来源：Energy Information Administration/Derivatives and Risk Management in Energy Industries, U. S. Department of Energy (2002), pg. 19.

按照从生产商到消费者以及为液化天然气单一世界市场（Lique fied Natural Gas，LNG）提供运输方式的管道基础结构，天然气世界贸易分为几个主要区域市场.美国是世界上最大的管道天然气市场.2000年，美国生产了19.3万亿立方英尺天然气，但消费了23万亿立方英尺.供给缺口被来自加拿大的3.2万亿立方英尺和来自世界市场LNG的0.5万亿立方英尺弥补.欧盟国家生产了10.5万亿立方英尺，消费了16.2万亿立方英尺，供应缺口由俄罗斯进口和少量LNG进口弥补.2000年，俄罗斯是世界上最大的天然气生产商，生产了20.3万亿立方英尺，其次为美国和加拿大.LNG的主要出口国是印度尼西亚、马来西亚、澳大利亚、卡塔

尔、尼日利亚和特立尼达岛. 日本是 LNG 最大的进口国.

天然气就像电力一样，所有供给者和使用者都为了这种商品而通过物理分布系统连接起来，从这种意义上来看，天然气是网络产业. 在美国境内，运输天然气的管道产业缺乏竞争. 图 7.20 显示了主要管道提供的地点和几个现货市场（定价点），它们出现在主要转运点（中心).

天然气和电力的区域套利同原油一样不起作用. 因为天然气管道和电力线路没有竞争者，失望的顾客不能购买供应商"其他系统". 另外，要实现竞争性运输定价是困难的. 因此，运输费用在非竞争性市场中被设定，结果是在局部区域，跨市场的任意价差可保持或多或少的独立性，而不是基于边际成本.

7.15.2 天然气现货价格

存在大量的基本因素驱动复杂的天然气价格行为，包括提炼、储存、运输、天气变化、政策、技术进步等. 图 7.21 列举了日天然气现货价格图，从 1992 年 1 月 2 日到 1999 年 12 月 30 日，在路易斯安那的亨利中心，它们是用美元每百万英国热单位（MMBTV）衡量.

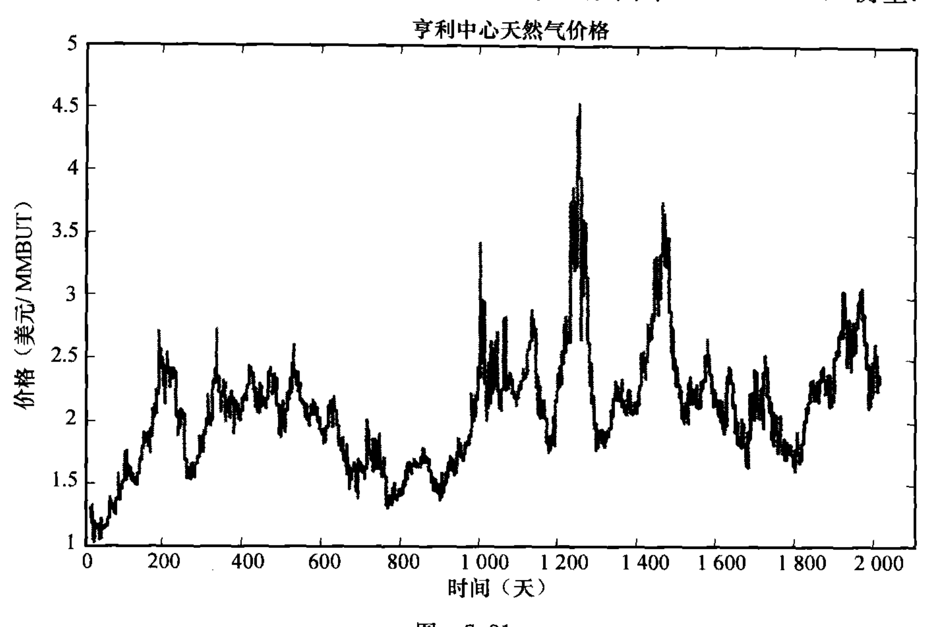


图 7.21

资料来源：Xu(2004).

一些重要属性与现货价格的行为有关. 像电力价格一样，天然气价格也是均值回复的. 另外，当价格上下波动频繁时，长期观点认为它们实际上围绕均值水平振动，是均值回复的效应.[46]天然气的均值回复似乎与市场对一些事件（如洪水、夏季热浪和其他会导致市场新增未预期到的供求失衡的新闻事件）的反应相关. 供应方的修正来匹配需求方，或者是事件的实际损耗（如天气回复到它们的平均季节水平）可能会导致天然气价格回复到它们的平均水平.[47]均值回复的速度取决于供求能够多快地回复到均衡状态或事件的影响消散. 围绕长期均值的价格振动取决于生产的成本和需求的正常水平.[48]

季节性同样是天然气价格的一个重要属性. 季节性主要源于正常需求波动，主要由天气相

关因素驱动. 正如我们所知，天然气主要作为住宅和办公楼取暖燃料，并且它也逐渐成为发电燃料. 寒冷冬季的天气导致了天然气高于平均消费量，然而在一些炎热的夏季，发电的需求也会增加，以维持空调和其他制冷设备运行.[49] 另一方面，"储存困难和运输能力的限制使得供应方缺乏弹性，不能及时跟上需求方的突然增长".[50]

7.16 天然气定价模型

我们能够像对电力价格建模一样对天然气价格建模. 考虑到天然气现货价格是季节项 $f(t)$ 和非季节状态项 X_t 之和，当长期均值和时间依赖波动率不变时，X_t 遵循单因素均值回复过程.

7.16.1 单因素模型

根据 Xu(2004)，令

$$S_t = f(t) + X_t \tag{7.132}$$

其中，

$$f(t) = bt + \sum_{i=1}^{N} \beta_i \cos\left(\frac{2\pi it}{365}\right) + V_i \sin\left(\frac{2\pi it}{365}\right) \tag{7.133}$$

和

$$\mathrm{d}X_t = \alpha(L - X_t)\mathrm{d}t + \sigma(t)X_t^r \mathrm{d}W_t \tag{7.134}$$

其中，

$$\sigma(t) = \exp\left(c + \sum_{j=1}^{K} \lambda_i \cos\left(\frac{2\pi it}{365}\right) + \omega_j \sin\left(\frac{2\pi jt}{365}\right)\right)$$

这里 W_t 是标准布朗运动，$r=0$，$\frac{1}{2}$或 1，并且 b，β'_i，v'_i，α，L，c，λ'_j和ω'_i全是常数. 为了保证为正，波动项 $\sigma(t)$ 为指数形式. $f(t)$ 中的线性项用来捕捉价格的趋势. $f(t)$和$\sigma(t)$中的正弦函数和余弦函数使它们可以随着季节变换而周期性地上下移动. 进一步说，对应于$N=1$ 和 $K=1$的三角函数可以捕获年度季节性，因为这些函数的周期都是 365（一年的天数）；对应于 $N=2$ 和 $K=2$ 的三角函数可以捕获半年度季节性，因为这些函数的周期是半年.

Xu(2004) 用从 1992 年 1 月 2 日至 1999 年 12 月 30 日的亨利中心价格使用最大似然估计估算没有季节性的单因素模型参数，如表 7.6 所示.

表 7.6

	模型 1 $r=0$	模型 2 $r=1/2$
α	0.018 1	0.013 9
L	2.144 1	2.152 6
σ	0.087 0	0.054 9

资料来源：Xu(2004).

7.16.2 双因素模型

考虑用一个 SDE 系统描述这些模型：

$$\begin{cases} \mathrm{d}S_t = \alpha(L_t - S_t)\mathrm{d}t + \sigma S_t^r \mathrm{d}W_t^1 \\ \mathrm{d}L_t = \mu(\gamma - L_t)\mathrm{d}t + \tau L_t^r \mathrm{d}W_t^2 \end{cases} \tag{7.135}$$

其中，α，σ，μ，γ 和 τ 是常数，$r=0$，$\frac{1}{2}$或 1. 我们假定维纳过程 $\mathrm{d}W_t^1$ 和 $\mathrm{d}W_t^2$ 是不相关的. 在

这个双因素模型中，天然气现货价格长期均值是遵循它自身均值回复过程而不是几何布朗运动的随机变量. 这个模型中能加入季节性. 遵循双因素均值回复过程的天然气现货价格可以被看成季节项 $f(t)$ 和非季节状态项 X_t 之和.

数学上，令

$$S_t = f(t) + X_t$$

其中，

$$f(t) = bt + \sum_{i=1}^{N} \beta_i \cos\left(\frac{2\pi it}{365}\right) + \eta_i \sin\left(\frac{2\pi it}{365}\right)$$

X_t 遵循均值回复过程，描述如下：

$$\begin{cases} dS_t = \alpha(L_t - S_t)dt + \sigma(t)S_t^r dW_t^1 \\ dL_t = \mu(\gamma - L_t)dt + \tau L_t^r dW_t^2 \end{cases}$$

其中，

$$\sigma(t) = \exp\left(c + \sum_{j=1}^{K} \lambda_i \cos\left(\frac{2\pi it}{365}\right) + \omega_j \sin\left(\frac{2\pi jt}{365}\right)\right)$$

在前面方程中，W_t^i 是维纳过程，$r=0$，$\frac{1}{2}$ 或 1，并且 b，β_i'，v_i'，α，L，c，λ_j' 和 ω_i' 全是常数，N 和 K 是正整数.

为了对天然气期货合约定价，在风险中性测度下，公式（7.135）的期望为：

$$\begin{cases} dS_t = \tilde{\alpha}(L_t - S_t)dt + \sigma\sqrt{S_t}dW_t^1 \\ dL_t = \tilde{\mu}(\gamma - L_t)dt + \tau\sqrt{L_t}dW_t^2 \end{cases}$$

其中 $\tilde{\alpha}$，$\tilde{\mu}$ 和 $\tilde{\gamma}$ 是风险中性参数，并且假定平方根过程（$r=1/2$）. 那么可以发现：

$$\begin{cases} d\overline{S}_t = \tilde{\alpha}(\overline{L}_t - \overline{S}_t)dt \\ d\overline{L}_t = \tilde{\mu}(\tilde{\gamma} - \overline{L}_t)dt \end{cases} \tag{7.136}$$

其中 $\overline{S}_t$ 和 $\overline{L}_t$ 分别是 S_t 和 L_t 在时刻 t 的期望值. 对于 $\overline{L}_t$，在边界条件 $t=t_0$，$\overline{L}_t = L_{t_0}$ 下，解第 2 个常微分方程，可得：

$$\overline{L}_t = (L_{t_0} - \tilde{\gamma})e^{\tilde{\mu}(t_0-t)} + \tilde{\gamma} \tag{7.137}$$

当边界条件是 $t=t_0$，$\overline{S}_t = S_{t_0}$ 时，对于 $\overline{S}$，将公式（7.136）代入公式（7.135）中的第一个常微分方程，我们得到

$$\begin{aligned} \overline{S}_t &= S_{t_0}e^{\tilde{\alpha}(t_0-t)} + \frac{\tilde{\alpha}}{\tilde{\alpha}-\tilde{\mu}}(L_{t_0} - \tilde{\gamma})(e^{\tilde{\mu}(t_0-t)} - e^{\tilde{\alpha}(t_0-t)}) - \tilde{\gamma}(1 - e^{\tilde{\alpha}(t_0-t)}) \\ &= e^{\tilde{\alpha}(t-T)}S_{t_0} + \frac{\tilde{\alpha}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t_0-t)} - e^{\tilde{\alpha}(t_0-t)})L_{t_0} + \frac{\tilde{\mu}\tilde{\gamma}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t_0-t)} - 1) - \frac{\tilde{\mu}\tilde{\gamma}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t_0-t)} - 1). \end{aligned}$$

因为 $F^{\tilde{\theta}}(t, T, S_t) = \overline{S}_t$，并且在初始时间 $t_0 = t$，$\overline{S}_{t_0} = S_t$ 以及 $\overline{L}_{t_0} = L_{t_0}$，在风险中性测度下，天然气期货价格为：

$$\begin{aligned} F^{\tilde{\theta}}(t,T,S_t) = {} & e^{\tilde{\alpha}(t-T)}S_t + \frac{\tilde{\alpha}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t-T)} - e\tilde{\alpha}(t-T)L_t \\ & + \frac{\tilde{\mu}\tilde{\gamma}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t-T)} - 1) - \frac{\tilde{\mu}\tilde{\gamma}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t-T)} - 1) \end{aligned} \tag{7.138}$$

如果考虑季节性，则期货价格为：

$$F^{\tilde{\theta}}(t,T,S_t) = f(T) + e^{\tilde{\alpha}(t-T)}(S_t - f(t)) + \frac{\tilde{\alpha}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t-T)} - e\tilde{\alpha}(t-T)L_t$$
$$+ \frac{\tilde{\mu}\tilde{\gamma}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t-T)} - 1) - \frac{\tilde{\mu}\tilde{\gamma}}{\tilde{\alpha}-\tilde{\mu}}(e^{\tilde{\mu}(t-T)} - 1) \quad (7.139)$$

7.16.3 校准

为了标定和估计模型的参数，可以使用最大似然估计. 校准（Calibration）是匹配从市场观测到的信息的一个必要过程，所以信息匹配的程度越高，就得越精确. 季节性可以从天然气现货价格和期货价格中推断出来. 为了揭示季节项 $f(t)$，很自然的一个观点是同时运用现货价格和期货价格的信息.

7.16.4 单因素模型校准

$\{S_t\}_{t=1}^n$ 表示历史现货价格，$\{F_{t,T_{ti}} | t=1, 2, \cdots, n; i=1, 2, \cdots, m\}$ 是期货价格，其中 T_{ti} 是 t 后的第 i 个交割，m 是在时刻 t 观测到的期货价格的数目. 这是可以从真实市场获得的数据. 另一方面，假定现货价格遵循方程（7.132）～(7.134）描述的过程，对于有季节性的单因素模型，我们有如下给定的理论期货价格函数 $F^{\tilde{\theta}}(t, T, S_t)$：

$$F^{\tilde{\theta}}(t,T,S_t) = e^{\tilde{\alpha}(t-T)}(S_t - \tilde{L} - f(t)) + \tilde{L} + f(T) \quad (7.140)$$

其中，t 是观测时间，T 是交割时间，$\tilde{\theta}$ 是活跃参数的集合，即 $[b, \beta_1, \beta_2, \cdots, \beta_N, \eta_1, \eta_2, \cdots, \eta_N, \tilde{\alpha}, \tilde{L}]$.

为了匹配模型的真实市场数据，我们需要找到一些合适的参数使得理论期货价格和实际期货价格越接近越好. 如果两个向量的距离定义在欧式空间，则对于 $t=1, 2, \cdots, n$；$i=1, 2, \cdots, m$，$\tilde{\theta}$ 的估计可以通过最小化 $F^{Q}(t, T_{ti}, S_t)$ 和 $F_{t,T_{ti}}$ 之差的平方和来获得. 特别地，我们需要求解：

$$\tilde{\theta} = \arg\min_{\tilde{\theta}} \sum_{t=1}^{n}\sum_{i=1}^{n}(F^{\tilde{\theta}}(t,T_{ti},S_t) - F_{t_t,T_{ti}})^2 \quad (7.141)$$

对于有季节性（$r=0$）的单因素均值回复模型，Xu(2004) 用仿造数据和真实数据（来自 1992 年 1 月 2 日至 1999 年 12 月 30 日的亨利中心价格）通过最小化式(7.141) 估计了风险中性参数，如表 7.7 所示.

表 7.7

	仿造数据			真实数据
	数值	估计的平均值	估计的标准差	估计值
$\tilde{\alpha}$	0.007 0	0.007 0	1.456 9e−13	0.007 3
$\tilde{L}$	1.700 0	1.700 0	1.885 9e−11	1.666 3

资料来源：Xu(2004).

对模型（7.133）（对于 $r=0$，1/2 和 1)，估计季节参数由表 7.8[51] 给定，其中 0 用来表示小于 1e−10 的一些值.

表 7.8

	仿造数据			真实数据		仿造数据			真实数据
	数值	估计的平均值	估计的标准差	预估值		数值	估计的平均值	估计的标准差	估计值
b	0.000 3	0.000 3	0	0.000 3	η_1	−0.050 0	−0.050 0	0	−0.046 8
β_1	0.150 0	0.150 0	0	0.163 9	η_2	−0.030 0	−0.030 0	0	−0.029 5
β_2	0.050 0	0.050 0	0	0.055 8					

资料来源：Xu(2004).

7.16.5 双因素模型校准

双因素模型校准是单因素模型校准的一个扩展. 首先，通过匹配实际期货价格和理论期货价格，我们“挖掘”潜在的因素，包括季节项 $f(t)$ 和长期均值回复因素 L_t.[52] 在这个过程中，对于 $f(t)$ 和一些风险中性参数，可以获得 X_t. 其次，当 X_t 和 L_t 已知时，运用最大似然（ML）方法通过这两个随机过程获得参数.[53]

为了从期货价格找到潜在的因素 L_t 和季节项 $f(t)$，令

$$A_i = f(T_{ti}) + e^{\widetilde{\alpha}(t-T_{ti})}(S_t - f(t)) + \frac{\widetilde{\mu}\widetilde{\gamma}}{\widetilde{\alpha}-\widetilde{\mu}}(e^{\widetilde{\mu}(t-T_{ti})} - 1)$$

和

$$B_i = \frac{\widetilde{\alpha}}{\widetilde{\alpha}-\widetilde{\mu}}(e^{\widetilde{\mu}(t-T_{ti})} - e^{\widetilde{\alpha}(t-T_{ti})})$$

然后通过方程（7.138)，我们有期货价格函数 $F^{\widetilde{\theta}}(t, T_{ti}, S_t) = A_i + B_i L_t$. 值得注意的是，期货价格函数不仅涉及 $[\widetilde{\alpha}, \widetilde{\mu}, \widetilde{\gamma}]$ 和 L_t，同样涉及 $f(t)$；因此，我们能够通过匹配期货价格全部得到它们.

因此，我们令 $\widetilde{\theta} = [\widetilde{\alpha}, \widetilde{\mu}, \widetilde{\gamma}, b, \beta_1, \cdots, \beta_N, \eta_1, \cdots, \eta_N]$. 和以前一样，$L_t$ 可以定义为 $\widetilde{\theta}$ 的函数：

$$L_t(\widetilde{\theta}) = \frac{\sum_{i=1}^{m}(B_i F_{t,T_{ti}} - A_i B_i)}{\sum_{i=1}^{m} B_i^2} \tag{7.142}$$

接着，解除 $\widetilde{\theta}$ 的限制通过下面方程获得最优的一个：

$$\widetilde{\theta} = \arg\min_{\widetilde{\theta} \in \mathbb{R}^{n_0}} \sum_{t=1}^{n}\sum_{i=1}^{n}(F^{\widetilde{\theta}}(t, T_{ti}, S_t, L_t(\widetilde{\theta})) - F_{t,T_{ti}})^2 \tag{7.143}$$

其中，n_0 是 $\widetilde{\theta}$ 的向量长度. 当 $\widetilde{\theta}$ 是已知时，$\{L_t\}_{t=1}^{n}$，$\{f(t)\}_{t=1}^{n}$ 和 $\{X_t\}_{t=1}^{n}$ 都是容易计算的.

表 7.9 展示了式(7.138) 的用亨利中心价格的仿造数据和真实数据的风险中性参数估计.

表 7.9

	仿造数据			真实数据估计值
	数值	估计的平均值	估计的标准差	
$\widetilde{\alpha}$	0.0110	0.0110	0	0.0111
$\widetilde{\mu}$	0.0020	0.0020	0	0.0018
$\widetilde{\gamma}$	2.0000	2.0000	0	2.0028

资料来源：Xu(2004).

对于亨利中心实际价格数据，图 7.22 展示了基于最小化双因素模型之差的平方和的真实和估计季节项是（7.143).

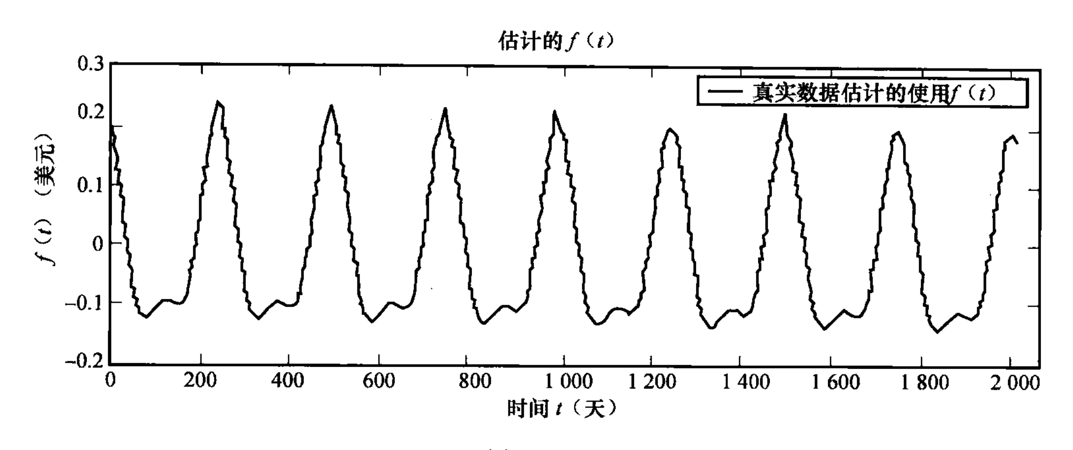


图 7.22

资料来源：Xu(2004).

图 7.23 显示了模型使用真实数据的估计现货价格 X_t 和长期均值回复参数 L_t.

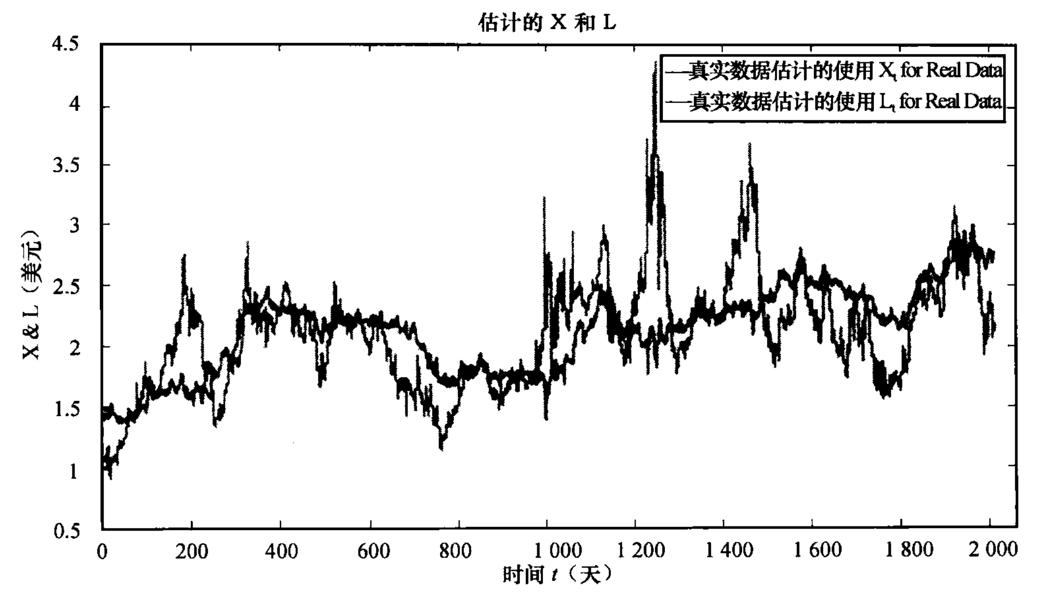


图 7.23

资料来源：Xu(2004).

使用 1992 年 1 月 2 日到 1999 年 12 月 30 日亨利中心的仿造数据和真实数据，Xu(2004) 估计了（7.134）中模型的参数.当公式（7.134）中的 $r=0$ 时，参数由表 7.10 给定.

表 7.10

	仿造数据			真实数据		仿造数据			真实数据
	数值	估计的平均值	估计的标准差	估计值		数值	估计的平均值	估计的标准差	估计值
α	0.024 0	0.025 2	0.004 5	0.024 4	λ_1	0.400 0	0.402 1	0.020 3	0.410 5
c	−2.600 0	−2.602 5	0.018 0	−2.561 6	λ_2	0.080 0	0.077 7	0.026 7	0.079 0

（续）

	仿造数据			真实数据		仿造数据			真实数据
	数值	估计的平均值	估计的标准差	估计值		数值	估计的平均值	估计的标准差	估计值
ω_1	−0.200 0	−0.798 3	0.020 2	−0.498 2	τ	0.030 0	0.029 9	0.000 4	0.032 9
ω_2	−0.070 0	−0.068 3	0.023 1	−0.066 0	γ	2.300 0	2.283 3	0.129 7	2.286 0
μ	0.005 0	0.006 8	0.003 0	0.004 8					

资料来源：Xu(2004)．

当 $r=1/2$ 时，参数由表 7.11 给定.

表 7.11

	仿造数据			真实数据		仿造数据			真实数据
	数值	估计的平均值	估计的标准差	估计值		数值	估计的平均值	估计的标准差	估计值
α	0.022 0	0.023 0	0.004 8	0.022 1	ω_2	−0.030 0	−0.033 3	0.022 4	−0.031 7
c	−3.000 0	−3.003 0	0.016 9	−2.989 4	μ	0.005 0	0.007 5	0.003 7	0.005 0
λ_1	0.380 0	0.378 7	0.021 2	0.379 1	τ	0.023 0	0.022 8	0.000 4	0.023 0
λ_2	0.075 0	0.074 3	0.019 8	0.074 8	γ	2.300 0	2.286 6	0.141 9	2.279 3
ω_1	−0.180 0	−0.182 8	0.022 9	−0.182 6					

资料来源：Xu(2004)．

对于所有三个模型$\left(\text{如 } r=0,\ \frac{1}{2}\text{和 } 1\right)$，季节因素的参数显示在表 7.12 中.

表 7.12

	仿造数据			真实数据		仿造数据			真实数据
	数值	估计的平均值	估计的标准差	估计值		数值	估计的平均值	估计的标准差	估计值
b	−0.000 1	−0.000 1	0	−1.447 6e−5	η_1	−0.050 0	−0.050 0	0	−0.052 7
β_1	0.150 0	0.150 0	0	0.148 3	η_2	−0.030 0	−0.030 0	0	−0.029 2
β_2	0.060 0	0.060 0	0	0.057 4					

资料来源：Xu(2004)．

图 7.24 展示了使用季节性的双因素模型进行实际和估计期货价格匹配（通过最优化(7.143)）的天然气期货拟合.

在中间的图中，粗曲线是实际期货曲线；细曲线是由（7.138）给定的期货函数图. 在下面的图中，粗曲线是实际期货曲线；细曲线是公式（7.134）模拟的 S_t 的 250 条路径的值，其中显示在表 7.5 中的风险中性参数是插入的，$r=1$. 为了比较，上面的图显示了季节项 $f(t)$（实线）和季节波动率函数 $\sigma(t)$（虚线）.

为了分析长期行为并预测天然气价格（还有石油和煤炭价格），可以使用价格演化的随机动态性，参见 Pindyk(1998). Pindyk(1998) 使用卡尔曼（Kalman）滤波方法估计了各种模型. 根据可耗竭资源生产理论和使用过去一个世纪实际价格的真实表现定价，Pindyk 显示非结构性模型应该加入均值回复到随机波动趋势线，反映不可观测的长期（总）边际成本.[54]

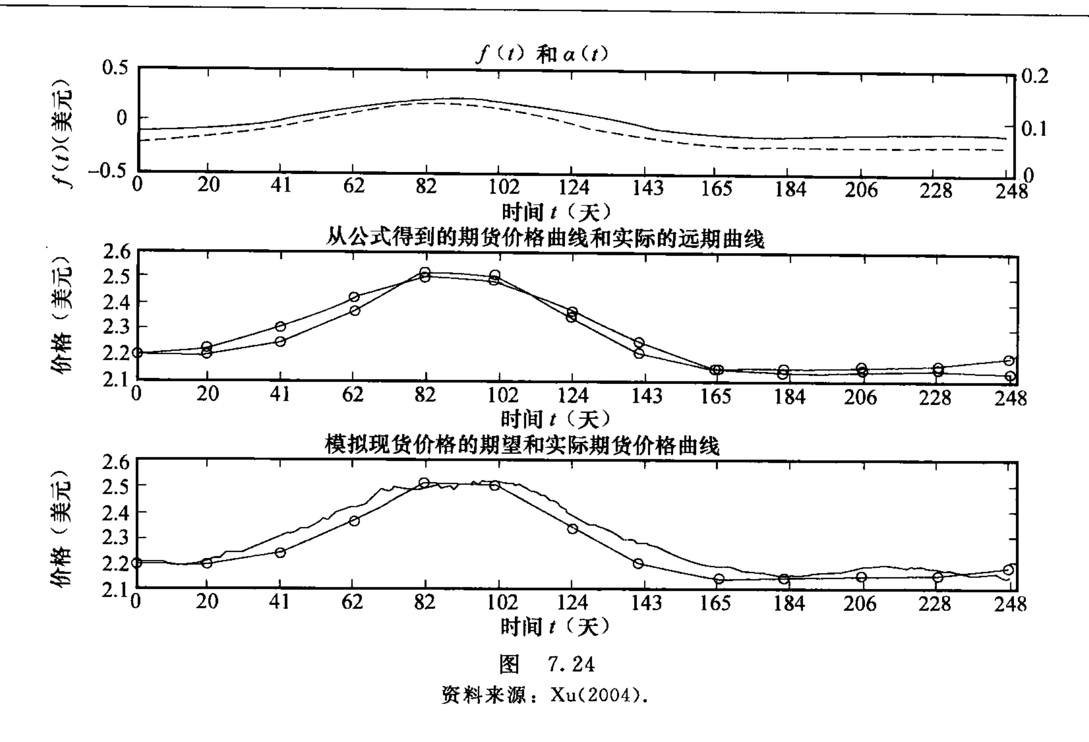


图 7.24
资料来源：Xu(2004).

7.17 应用 Matlab 定价天然气

对于使用实时天然气数据的天然气期货定价，由 James Xu(2004) 编写的关键 Matlab 代码的完全列表可参阅附录 B.

7.18 天然气与电力互换

当市场的一方或者双方面对一个与参考价格不同的现货市场价格，基差风险就会产生. 为了管理这个风险，远期和期货合约就不能用了，因为它们仅仅在统一市场价格确定的情形下起作用. 然而，我们有可能需要像基本合约那样的其他衍生品合约工具来管理生成的基差风险. 基差互换允许一个人锁定局部的固定价格而不是期货合约的交割点. 这个可以或者作为一个实物交易或者作为一个金融交易被完成. 最广泛使用的天然气期货市场要求在路易斯安那的亨利价格中心交付. 在 OTC 市场，基差合约可以用来对冲局部、生产甚至是期货交易所标准合约和合约使用者的特殊情况的临时差异. 例如，假设在田纳西的当地供销公司（LDC）能够与天然气生产者参与互换合约，并使用亨利中心价格作为参考价；然而，如果当地现货价格与亨利中心价格不一样，当地供销公司将会失去价格确定性，正如图 7.25 所示.

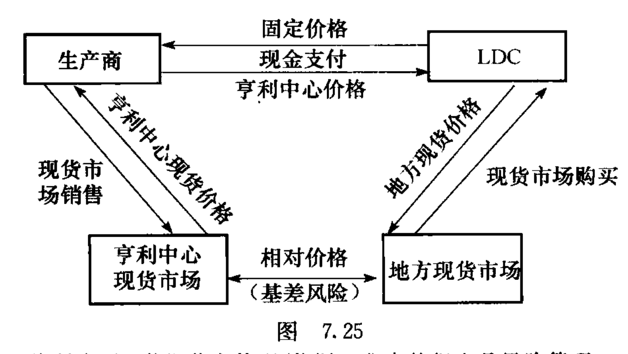


图 7.25
资料来源：能源信息管理/能源工业中的衍生品风险管理，美国能源部（2002）.

在这个例子中，当亨利中心价格高于田纳西价格超出互换合约的初始值，LDC 获益，因

为生产者的支付将超过在当地市场购买天然气的价格.[55]实际上，LDC公司将比支付给生产者每千立方英尺的固定金额支付得更少.相反，如果田纳西价格较低，生产者的付款将无法覆盖当地市场LDC的天然气账单.在一个天然气基差互换中，OTC交易者将支付给LDC田纳西价格与亨利中心价格（对于天然气的名义账户）之间的差额交换固定款项.各种合同条款是无限制的.例如，弹性支付可以被定义为"日或者月度平均（加权或者未平均的）价差；可被设置上限；或者当合约最高价格被超过的时候，它可能需要LDC来分担成本."[56]该OTC合约用于取消亨利中心价格和LDC当地现货市场价格差，使得LDC实现价格确定性.

仅当"支付的基差差异——对财务费用和资金的时间价值按时间进行平均和调整——低于来自于LDC的固定支付"[57]提供基差合约的交易员才能生存.来自OTC交易员的竞争仅能减少供应基差保护的溢价.减少波动价格差异的潜在原因将"需要更多管道能力、更多储藏能力、基于成本运输定价，以及交割系统本身的其他物理和经济变化."[58]

类似地，在电力期货市场，大多数公司在交割地有价格暴露，如COB、Palo Verde和PJM Interconnect等地方.因此，在其他地方使用纽约商品交易所期货合约管理价格风险的公司会暴露出基差风险——在不同地点的价格差异.交易员和公司可以使用基差互换和合约来对冲他们的风险.基差市场逐步发展，使得公司可在美国和加拿大的主要交易地对冲风险.这些基差市场流动性很好，具有狭小的买卖单价差（通常小于0.02美元，但在波动时期可能更宽）和大成交能力.基差市场同样开始在美国西部的电力市场发展（例如COB、Paolo Verde和Mid-Columbia).基差市场不需要运输管线和管道的实物连接.例如，存在一个Sumas天然气基差市场，甚至很难将天然气从该地运输到亨利中心.根据Stoft、Belden、Goldman和Pickle(1998）的研究，我们列举适用不同用户的金融交易来展示这些基差互换的机制.

7.18.1 发电厂

为了锁定丹佛市的电价，发电厂可以卖出期货合约（或者价格互换）和基差互换（见图7.26).假定发电厂以18美元/MWh的价格售出期货合约，6个月之后交付，同时同意在支付现货价格以换取COB价格和溢价的基础上售出基差互换.在未来的6个月内，发电厂以丹佛电价现货价格（B）出售电量，按这一价格（B）支付给基差合约方，并从合约方收到COB现货价格（A）和固定溢价，同时以现货价格A购买期货合约.所有这些交易相互抵消，基于最初的期货合约，发电厂最终将会获得18美元/MWh的收益和合约方支付给它的溢价.上述描述的是一个完整的金融交易过程，实体

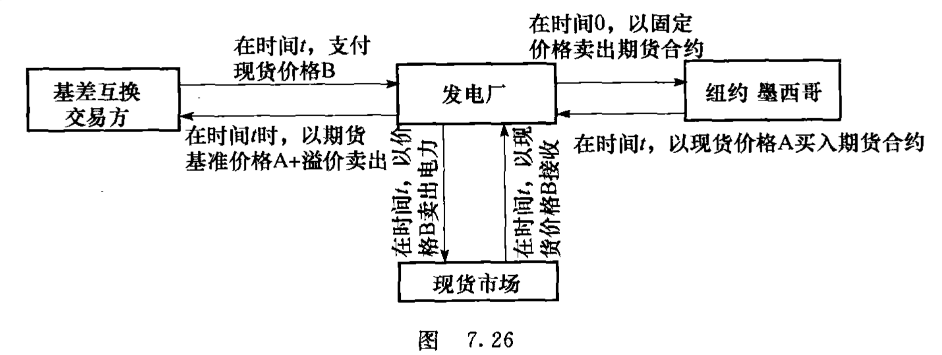


图 7.26

资料来源：Stoft S，Belden T，Goldman C和Pickle S(1998)，35.

交易同样是可操作的，这里发电厂向基差互换合约方提供电力，获取的收益是COB现价和溢价.

在这个基准互换交易中，存在的风险是发电厂可能不会以COB期货价格来购买期货合约，我们可以通过价格互换来更好地规避风险.这样，发电厂将支付给价格互换交易方道琼斯COB指数均值，从而抵消基差交易方的道琼斯COB指数均值.[59]

7.18.2 最终用户

为了锁定丹佛市电价，最终用户可以购买期货合约（或价格互换）和一个基差互换（见图7.27).假定最终用户以18美元/MWh的价格买入期货合约，并且同意支付COB现货价格和溢价作为对丹佛市变动的现货市场价格的回报，用户可以选择通过实体交易或金融交易来履约。无论发生何种情况，用户都可以锁定18美元/MWh的电价和溢价.用户也可以选择通过价格互换（而不是期货合约）的方式来锁定丹佛市或其他地方的价格.

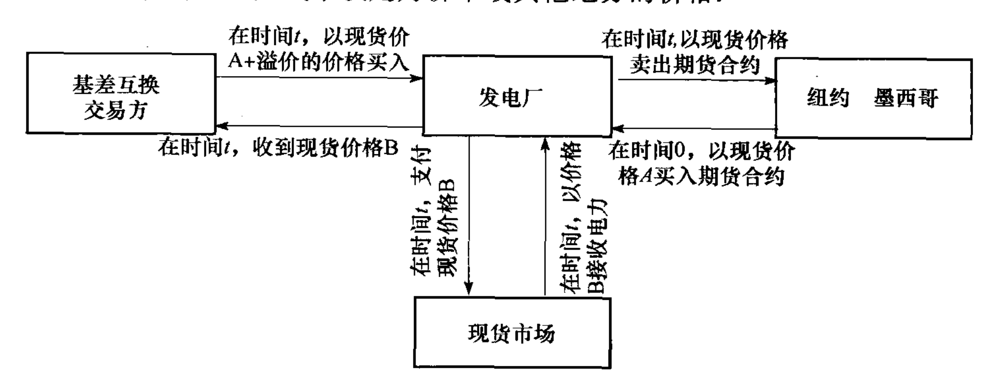


图 7.27
资料来源：Stoft S，Belden T，Goldman C 和 Pickle S(1998)，35.

在价格互换交易中，买方所同意支付的固定价格是在交易时双方谈判达成一致的价格，收到的价格等于约定月份发布在《华尔街日报》上的虚盘COB指数价格的简单平均.[60]尽管理论上并没有约定互换交易的规模，但是通常最多以25兆瓦增量交易.因为在美国西部，每日的高峰负荷时间是16小时，从上午6点到晚上10点，一周6天（周一到周六).因此全国电力总容量等于兆瓦数乘以当月的日数（除去周日和节假日）再乘以16.同期货合约一样，当价格上升时，价格互换买方的利润增加，反之亦然.当道琼斯COB指数平均值超出固定价格时，互换交易买方（固定价格的支付者）将从交易中获利；反之，若均值低于固定价格，则卖方获利.互换交易可以用来保值或投机.

例如，为了锁定1998年7月电价，某用户购买了价格互换（见图7.28).假定该用户同意支付25美元/MWh，收益是道琼斯COB价格均值.如果2006年6月的COB现货价格是20美元/MWh，那么用户将会在现货市场以20美

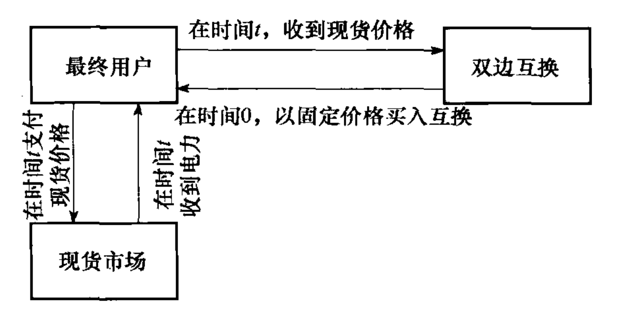


图 7.28
资料来源：Stoft S，Belden T，GoldmanC 和 Pickle S (1998)，pg. 35.

元/MWh 买入电力，并将会从互换合约方拿到 20 美元/MWh，并且支付给互换合约方 25 美元/MWh. 通过互换交易，用户得到一个可保证价格 25 美元/MWh，但是若价格跌破 25 美元/MWh，那么他将无利可图.

市场专员可以代表发电厂和用户来执行这些交易，以确保他们在某个市场而不是 COB 或者 Palo Verde 得到一个固定的价格. 同样市场专员也可以在交易过程中扮演基差交易方的角色. 如果双方对交易过程中使用的金融工具感到不便，发电厂和用户可以通过市场专员来完成交易.

对于在电力市场和商品市场中使用互换交易和期货合约来套期保值和投机获利的研究，详见 Stoft、Belden、Goldman 和 Pickle(1998) 和 Schwartz(1997).

尾注

1. Lucia J 和 Schwartz E(2001)，2.
2. Id. 2.
3. Deng S(1999)，4.
4. Id. 2. 另外，如市场规则和市场结构等规章性问题同样对竞争性的电力市场行为有影响，因此，不同国家有差异.
5. Lucia J 和 Schwartz E(2001)，2.
6. 参见美国能源情报署（EIA）报告，年度能源展望(2002).
7. Doerr U(2003)，3.
8. Id. 3.
9. Lucia J 和 Schwartz E(2001)，13.
10. Lucia J 和 Schwartz E(2001)，15.
11. Id. 15.
12. Id. 15.
13. Lucia J and Schwartz E(2001)，20.
14. 北欧电力交易所创立于 1993 年 1 月，建成初期它仅覆盖挪威市场，现在它已成为自愿的跨国市场，包括 1996 年 1 月加入的瑞典，1998 年 6 月 15 日加入的芬兰和 1999 年 7 月 1 日加入的丹麦北部地区. 北欧电力交易所有两个市场，一个是“现货市场”(Elspot)，另一个是“金融市场”(Eltermin 和 Eloption)，同时提供清算服务. Elspot 是一个现货市场，前日电力合约的交易是为了下一天 24 小时的实物交割.

    现货市场的每个合约涉及一个负荷在给定的时间内，表示为兆瓦时，（1MWh 等于 1000kWh)，以及价格/MWh。这个价格称为系统价格它在第二日对每个小时是各自固定的，对于在全部市场区域（所谓的北欧电力交易区域）的所有参与者而言是基于供需平衡的，这没有考虑不同国家在网格中的容量限制（“瓶颈”）. 系统价格因而被定义为市场参与者交易在没有运输限制的整个交易区域电力的市场清算价格. 它同样被作为北欧电力交易所金融市场的结算参考价.
15. $S_t = \alpha + \beta D_t + \sum_{t=2}^{12} \beta_t M_{tt} + X_t$ 和 $\ln S_t = \alpha + \beta D_t + \sum_{t=2}^{12} \beta_t M_{tt} + Y_t$.

16. Lucia J 和 Schwartz E(2001)，20.
17. Doerr U(2003)，14.
18. Id.
19. Id. 14.
20. Id. 1.
21. Id. 1.
22. Id. 1.
23. Doerr U(2003)，21.
24. 参见 London(2004)，其中给出了应用于美国股票期权的 LSM 的 C++实现.
25. Doerr U(2003)，第 22 页.
26. 这些和至多有一个非零加数.
27. 参见 Doerr U(2003). 复述得到了作者的许可.
28. Doerr U(2003)，25.
29. Doerr U(2003)，26.
30. Doerr U(2003)，27.
31. 向上摆动的尖峰和向下摆动的尖峰被设定为相等.
32. Doerr U(2004)，34.
33. Deng S(1999)，24.
34. Id. 25.
35. Id. 25.
36. 参见 Deng(1999).
37. Deng S(1999).
38. Xiong L(2004)，26.
39. 复述得到了作者的许可.
40. 当 $t \to \infty$ 时，X_t 存在一个有限分布. 有限分布就是所谓的平稳分布，它的傅里叶变换是所谓的平稳特征函数（参见 Grimmett G 和 Stirzaker D(1982)).
41. 此处使用复合梯形规则来近似积分，$\sum_i^{l}(A_i)$ 表示 A_i 之和，A_i 的第一项和第二项减半.
42. Xiong L(2004).
43. 令 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 分别为 $K \times L$ 和 $M \times N$ 矩阵，它们元素的指标如下：

$$A[k,l], \quad k = 0,1,\cdots,K-1, l = 0,1,\cdots,L-1$$

$$B[m,n], \quad m = 0,1,\cdots,M-1, l = 0,1,\cdots,N-1$$

我们定义 Kronecker 乘积 $\boldsymbol{A} \otimes \boldsymbol{B}$ 为 $KM \times LN$ 矩阵：

$$A \otimes B = \begin{bmatrix} A[0,\ 0]\boldsymbol{B} & A[0,\ 1]\boldsymbol{B} & \cdots & A[0,\ L-1]\boldsymbol{B} \\ A[1,\ 0]\boldsymbol{B} & A[1,\ 1]\boldsymbol{B} & \cdots & A[1,\ L-1]\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ A[K-1,\ 0]\boldsymbol{B} & A[K-1,\ 1]\boldsymbol{B} & \cdots & A[K-1,\ L-1]\boldsymbol{B} \end{bmatrix}$$

其中，$(A \otimes B)[m+kM,\ n+lN] = A[k,\ l]B[m,\ n]$.

44. 1994年，NYMEX发布了裂解价差合约.在保证金需求方面，MYMEX将多个期货的裂解价差买卖看成一笔交易.裂解价差合约帮助提炼者同时锁定原油价格、燃料油价格和无铅汽油价格，以保证固定的提炼边际.裂解价差合约簇的一个类型是购买三个原油期货(30 000桶)，卖出两个一个月以后的无铅汽油期货（20 000 桶）和一个燃料油期货(10 000桶).3∶2∶1的比例近似于真实世界提炼产量的比率——从三桶原油中可以生产两桶无铅汽油和一桶燃料油.消费者和生产者都仅关注裂解价差合约的保证金需求.它并不包含标的交易的单独保证金（参见美国能源情报署（2002），22页).
45. 天然气是可燃物，是单烃的气态混合物，它通常被发现于由疏松岩石形成的深层水库下面.天然气是化石可燃物，几乎全由甲烷构成，但也含有一些微量其他气体，包括乙烷、丙烷、丁烷和戊烷.
46. Xu Z(2004)，7.
47. Xu Z(2004)，8.
48. Id.
49. Id. 8.
50. Id. 8.
51. 对于三因素模型来说$\left(r=0,\ \frac{1}{2}和\ 1\right)$，季节项估计是相同的，因为三个因素拥有由等式(7.139）给出的相同的期货价格函数.
52. Xu Z(2004)，80.
53. Id. 40.
54. Pindyk R(1998)，33.
55. 美国能源情报署（2002)，21.
56. Id. 21.
57. Id. 21.
58. Id. 21.
59. Stoft S，Belden T，Goldman C和Pickle S(1998)，40.
60. Stoft S，Belden T，Goldman C和Pickle S(1998)，35.

第 8 章　电力衍生品的定价：理论和 Matlab 实现

Craig Pirrong

休斯敦大学鲍尔经济学院金融学教授

8.1　引言

能源衍生品已经出现一段时间了. 20 世纪 70 年代引入了丙烷和燃料油的交易期货衍生品，20 世纪 80 年代开始原油期货交易，20 世纪 90 年代天然气期货出现. 有原油和天然气产品的领路，能源衍生品的场外交易市场也发展起来了. 一般的场外交易结构与利率或股票市场结构相似，也包括固定对浮动利率互换、远期和期权市场. 一些国外产品也在市场上交易，而且能源价格的变动正逐渐成为债务结构的一部分.

像石油互换这样的产品可以很容易地运用标准的套利理论进行估价，但是对能源期权进行估价时，建模非常困难. 这与股市、固定收益和货币市场的问题一致，甚至难度更高. 特别地，能源收益不能很好地服从正态分布. 能源收益不仅呈现过分的曲线峰态而且也经常倾斜. 随机波动和震荡也是常见的问题. 能源市场有着一些金融市场没有的复杂性. 更值得注意的是，季节性对一些能源产品来说是一个重要的考虑因素，天然气就是一个例子.

尽管很复杂，但在对可储存的能源衍生品建模时还是可以利用很多在股市、货币和固定收益市场发展起来的手段. 由于能源的不可储藏性，最新的能源交易并非易事，如电力衍生品. 大多数非常好用的标准估价技术是以布朗运动动态特性为基础的，然而目前，电价波动已经从布朗运动中区分开来，因此情况确实变得难于处理. 除此之外，尽管在金融和很多能源市场上数量风险非常有限甚至完全没有，但在电力市场却是一个很重要的风险. 数量风险是电力估价问题的一个重要考虑因素，而传统模型不能自然地将数量风险考虑进去.

因此，建模面临一个选择：是运用更复杂的价格形成过程（结合某些数量过程）还是利用将数量作为模型一部分的基于基本面的估价方法. 后者比较可行，因为电力市场的基本面非常显而易见. 天气、用电负荷、燃料价格和断电情况是电力价格的主要决定因素，这些因素都是可观察到的. 建模的主要困难是由于电力系统极其复杂多元. 建模者必须从这些复杂性中审慎提炼建立一个易于控制的，低维度的模型. 没有一个可控模型可以囊括现实市场的所有特点，关键是要有独到之处：一个巧妙简化的基于基本面的模型比一个根据以跳跃扩散为基础的价格波动的外生规定特征建立的定价理论要有效得多.

电力衍生品估价还面临另外一个问题. 电力市场本质上是一个不完整的市场. 甚至当我们使用一个外生规定价格形成过程的标准估价方法时，电力也不能被认为是一种资产. 因为电力不可储存，所以不可能执行动态交易策略. 因此，不能运用标准的风险对冲工具——只要我们选择不连续的电价运动过程，对冲工具就失去作用. 类似地，基本面模型中的很多价格决定因素，如天气、用电负荷和断电情况，也不是可贸易资产，所以无法直接对冲这些风险.

市场的不完整性意味着有必要估计相关风险的市场价格. 因此，模型校准是任何估价方法的必要组成部分.

本章随后部分会对这些问题进行详细说明. 8.2 节对电力市场进行简要介绍. 8.3 节分析电价可能的形成过程,分析表明在电力市场基于价格形成过程的估价方法是有问题的. 8.4 节开始关注基于基本面的模型,这些模型非常复杂,像 Eydeland-Wolyniec(2002) 的混合模型都过于复杂,需要一个更为简单的、维度更低的、能够抓住电价波动显著特征的、市场数据可观察到的估价方法. 本章后面部分主要就是建立这种更简易的模型. 8.5 节详细讨论了 Pirrong-Jermakyan 的基于基本面的模型. 8.6 节描述了如何根据观测到的市场数据来严格校准这个模型. 8.7 节和 8.8 节分析期权,并说明对电力期权进行估价的 Matlab 代码. 8.9 节讨论 PJ 模型对电力期权价格行为的意义. 8.10 节对本章进行简单总结.

8.2 电力市场

在美国和大多数其他发达国家,电力由传统的纵向一体化的公用事业单位提供,易受价格或收益率调整的影响,或者由国家垄断. 这些实体单位进行电力生产,然后通过高压线远距离传输,最后配送给在垄断服务区内的客户. 电力行业起步于 20 世纪 80 年代,在 20 世纪 90 年代迅猛发展,如今电力生产、运输和配送已经完成重组. 尽管不同国家之间以及美国不同地区之间的重组细节都不尽相同,但是大多数重组机制都具备一些相同的显著特征. 首先,纵向一体化已经大幅缩减,纵向一体化的企业在某种程度上已经被独立的生产、运输和配送企业取代. 服务地区的垄断已经被侵蚀了. 其次,批发和零售市场在重组后是必不可少的,而在纵向一体化的电力行业是不必要的. 因此,大多数重组后的电力行业存在批发市场,独立的电力生产商之间互相竞争,为负荷服务单位、工业电力用户和大的商业电力用户供电. 不仅如此,一些地方(如美国的得克萨斯州)已经在零售市场上展开竞争. 这样,零散电力用户就可以选择他们的家庭用电供应商.

在竞争性的电力批发市场的规划上存在很多差异. 一些市场很大程度上是双边的场外市场. 在这些市场中,电力生产商就资产运营进行独立决策,并与电力用户签订双边合同. 其他一些市场则更加正规和集中. 例如,美国的 PJM 市场是集中运营的日前和实时电力市场. 电力生产商发出要约,规定他们愿意提供的各种不同电量的价格,电力服务商也相应报出他们愿意购买的不同电量的价格. 将所有的供应绘制成供给曲线,报价绘制成需求曲线;供求曲线的交点确定市场结算价格;然后利用生产商们的供应来安排电力生产以使成本最小化.

在任何地方,行业重组都不会是一帆风顺的,加利福尼亚就出现过很多错误. 这些问题可以归因于电力作为一种商品的天然性质. 电力不能大量储存满足实际用途.[1] 而且在实际生活中,电力供应不足将导致用电管制,这样会造成巨大的经济损失. 因为电力不能储存,当需求猛增或电力生产商突然停止产电时,必须确保在任何时刻市场上都有充足的电量供应. 电力也是一种高度本土化的商品,输送限制意味着即使相距很近的地方,电力价格也可能出现巨大差异,而且这些价格差异在短时期内可能迅速发生变化.

不可储藏性意味着不能利用库存来缓解供需冲击的影响,其他商品也是如此,包括其他能源商品,如石油和天然气. 因为电力需求可能随着天气变化剧烈波动,电力供给也可能由于电力生产和输送的机器设备出现问题而发生波动,不能利用库存来缓解这种冲击意味着电力价格可能随供需的随机变化而大幅波动.

电力价格的极端波动（在下一节有详细说明和讨论）给市场参与者带来巨大风险.很多市场参与者，包括电力生产商和负荷服务单位，都易受数量风险的影响.这些风险带来对对冲工具的需求，这种工具在行业重组过程中已经发展起来了.对冲工具包括标准远期合约和多种期权.一些远期合约期限非常短，是在距交割期非常短的时间内签订的，比如说，在电力市场有很多合约是提前一天，甚至是提前一小时签订的.也有一些电力远期合约是在更长一段时期内交割，并在距交割期相当长一段时间签订.例如，有一种远期合约相当普遍，它是在一个月的高峰时段按比例进行电力交割的.除远期合约外，也有大量电力市场的期权合约，这会在 8.7 节中讨论.虽然已经有一些工具在交易所交易，但大多数电力衍生品都是在场外交易的.

尽管电力衍生品合约与其他能源衍生品合约表面看来很相似，但是电力的独特价格行为特征意味着对其他商品很好适用的估价方法用在电力估价上是有问题的.下一节将更加详细地讲述电力价格行为，并讨论运用传统估价方法给电力远期和期权定价存在的内在问题.

8.3 运用传统估价方法进行电力估价的问题

衍生品的传统定价方法是记录下衍生品合约中的资产或商品价格的随机过程.由于电力价格的极端非线性和季节性，这种方法在电力市场上的运用存在困难.电价的这些特点使得建立一个可控的并包含电价运动显著特征的简化的电价运动过程并不实际.

图 8.1 描述了 2001—2003 年 PJM 市场的每小时电价变动情况.该图显示了任何一个电力价格波动模型都必须解决的电价运动特征.构成 Black-Scholes 模型基础的线性扩散模型明显不能体现图 8.1 描述的价格运动特征；电价运动不遵循传统随机游走模型.电价围绕一个特定水平（大约 20 美元/MWh）上下波动，但偶尔也会向上跳跃，有时可达 1 000 美元/MWh.

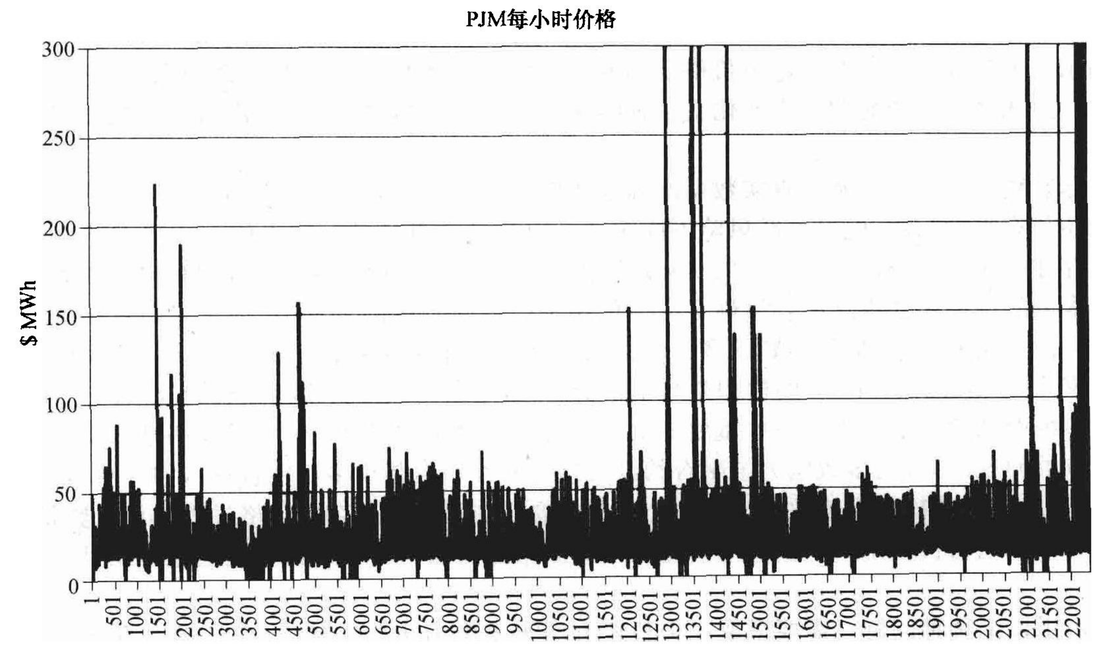


图 8.1 PJM 每小时价格

考虑到图 8.1 所示的电力价格的内在非线性，一些研究者提出包含跳跃因子的电价模型. 但这会出现其他问题. 例如，Merton(1973)提出的那个简单的跳跃模型是不适当的，因为在模型中跳跃因子的影响是永久的，而图 8.1 显示电力价格的跳跃会迅速逆转.

另外，传统跳跃模型中，价格要么向上跳跃要么向下跳跃，而在电力市场，价格先向上跳跃，随后很快下降. 为解决这些问题，Baŕz 和 Johnson(1999)引入均值回复和服从指数分布(因此为正）的跳跃因子. 然而这个模型假定大的电价冲击以与小的价格波动相同的速度衰减. 这在电力市场是很难让人信服的. Geman 和 Roncoroni(2006)建立了一个模型来缓解这种约束，但是这个模型对价格向上跳跃超过一个临界值是有条件的：a）价格向下跳跃的幅度与之前向上跳跃的幅度无关；b）下一个跳跃必须向下跳跃（即一旦价格突破临界值，那么之后的向上跳跃被排除在外). 而且在这个模型中，跳跃过程的剧烈程度不依赖于跳跃是否已经发生. 以上是这个模型所有问题的特点.

Barone-Adesi 和 Gigli（2002）尝试用一个严格的转换模型来把握现货电力价格的行为特征. 然而这个模型不允许连续的向上跳跃，向上跳跃之后必须是向下跳跃的约束使得这个模型不是马尔可夫过程. Villaplana(2004）通过规定一个价格运动过程来缓解这种约束，这个价格运动过程是连续和跳跃过程的加总，这两个过程的均值回复速度不同. 形成的新的价格运动过程不是马尔可夫过程，所以很难用来为衍生品合约估价.

跳跃型模型的估计也存在困难. 特别是一个合理的跳跃模型应该考虑价格及其跳跃强度和幅度的季节性，高需求时出现大幅跳跃的概率比低需求时大. 例如，根据美国需求的特征，在夏季很可能发生大幅跳跃. 用现有的有限的时间序列数据估计这种模型是个极大挑战. Geman 和 Roncoroni(2006)将这一特性考虑进去了，大多数其他的模型都没有做到. 而且由于这个问题有很高的计算强度，连 Geman 和 Roncoroni 都不能根据数据估计参数，而是在先验结果的基础上人为规定非齐次跳跃强度函数的参数. 严格转换模型的拟合也存在问题，尤其是当模型不是马尔可夫过程时，要符合电力价格运动的现实特征要求模型必须是马尔可夫过程（Geman, 2005). 除此之外，产能和需求变化也会影响跳跃的强度和幅度. 目前还没有哪个模型将此考虑进去.

尽管跳跃模型可以利用真实数据准确描述电价的运动特征，但作为电力衍生品合约的估价基础还是有很大问题. 电价跳跃风险不能对冲，因此电力市场是不完善的.[2] 一个允许多种跳跃幅度存在（跳跃大小连续为佳）的实际跳跃模型为了估价要求多种市场风险价格；跳跃幅度连续要求风险价格函数也连续，以确定与估价相关的风险价格的均衡值. 而且风险价格函数可能随着时间变化而变化. 估价问题的多元性使电力市场定价变得复杂得多. 事实上，现货价格模型越精密（Geman-Roncoroni 的模型是最精密的)，确定市场风险价格函数就越复杂.

传统的估价方法应用于一些重要的电力估价问题也非常困难，特别是那些结果受数量和价格影响的问题. 尽管大多数电力金融合约没有这一特性，但是很多现实合约（如负荷服务交易）有此特性. 在传统框架下给数量敏感性合约定价要求在已经很复杂的价格过程中加入数量过程. 此外，因为电力价格是关于负荷（发电量）的非线性函数，所以价格与发电量可能呈高度非线性关系.

由于以上种种问题，需要采取一个替代方法来给电力衍生品估价. 幸运的是存在一种替代方法. 该方法利用了决定电力价格的基本面因素非常明显这一事实，这与货币、股票和固定收

益市场的情况形成鲜明对比. 下一节详细介绍了基于基本面的模型.

8.4 基于基本面分析的模型

传统方法对电力估价不适用，那么它是否就完全没用呢？不是的. 关了一扇门就会开一扇窗. 在其他市场，如货币市场，基本的价格决定因素有很多并且很模糊，不能很确切地被观察到. 相反，在电力市场，基本面十分透明. 由于它的透明性，建立基于基本面的模型来给电力合约定价是可能的. 但并不表示这很容易. 基本面建模一定会遇到很多挑战，特别是电力市场的复杂性和不完善性，不过仍然是可行的.

本书介绍了两个基本且相关的基本面建模方案：Eydeland-Wolyniec（“EW”）和 Pirrong-Jermkyan（“PJ”）. 两者有一些共同点，但关键点不一样. 两者都利用非线性函数转换价格基本决定因素的线性过程，抓住了电力价格的非线性特征. EW 是一种试图把握所有基本面因素的模拟方法，而 PJ 是一种仅考虑几个关键因素的以偏微分方程为基础的方法. 更为重要的是，PJ 运用理论上很严格的方法根据观测到的市场价格来校准模型，而 EW 所使用的校准方法则更加灵活.

EW(2002) 描述了一种详细的基本面建模方法. 他们提出运用电力市场的多元模型来给电力价格建模，该模型试图把握电力市场的很多复杂特性. EW 方法里最重要的是它发现了电力价格的三个主要决定因素：产电使用的燃料成本、电力需求（受天气推动，用负荷直接衡量）和电力生产资源的有效性（例如，断电情况）. EW 认识到，在任何电力市场，电力生产都使用很多种燃料. 例如，在美国的大多数电力市场，有一些单位用煤，其他一些用天然气、核燃料或燃料油. 所以，EW 对市场上所有燃料的远期价格形成过程都作了详细说明. 这又反过来要求描述所有这些燃料的相关性.

EW 认识到产电设备的多样性意味着不同产电单位的断电概率不同，断电期限不同，还有规划好的断电形式也不同. 他们的模型将这点考虑进去了，不同单位使用不同的断电概率.

利用发电厂的物理性质——最重要的是耗热率，通过耗热率可以知道生产 100 万瓦的电需要消耗多少燃料——燃料价格和发电厂的资源利用率，EW 可以计算出特定市场上生产一定量电力的边际成本. 描述产电量与边际成本两者之间关系的曲线被普遍称为*发电堆*(generating stack). 然而 EW 发现电力生产商愿意出售电力的价格可能会与边际成本出现分歧. 因为是电力生产商的报价决定现货价格，而不是由边际成本决定，EW 认为市场供给曲线（报价堆），与发电堆不同，但是这两条曲线之间有稳定关系，由 EW 模型的三个参数（α 参数）概括.

这三个参数在校准过程中也起着重要作用. EW 不依赖任何历史数据，而是主张利用可观测到的远期和期权价格来校准模型. α 参数在 EW 模型框架里起着举足轻重的作用. 校准过程一石二鸟. 从字面上理解，指的是 α 参数将实际决定价格的报价堆和可以通过观测到的数据计算得到的发电堆联系起来了. 尽管 EW 不强调市场不完善性，但还是将其考虑进去了. 在任何市场，由于市场风险价格的存在，远期价格可能会与预期的未来现货价格不一致. 选择合适的 α 系数使得模型估计的预期价格与市场价格有效匹配，然后调整概率测度计算期望值.

由于它的多元性，要应用 EW 模型就必须用蒙特卡罗方法. 我们知道燃料价格、温度和断电情况都是不断变化的，有了电力价格的所有决定变量动态性的详细说明，就可以模拟这些变

量. 将变量模拟值代入校准后的报价堆可以得到现货电力价格. 从而有可能确定随现货价格变化而变化的合约价值. 模拟得到的平均价值（适当贴现）就是对合约价值的估计.

EW 模型抓住了一些电力市场更复杂的特性，尽管这种方法的多元性很具吸引力，但代价也很高. 例如，它必须估计大量变量之间的相关性，但经常缺少数据. 此外，模型校准计算量很大. 首先必须选择一个 α 值，根据所有的合约报价模拟得到结果，然后确定每个合约价格对参数变化的敏感性，再用这些敏感性调整参数进行另一次模拟，如此反复，直到实现收敛. 反复进行蒙特卡罗模拟不恰当，应该尽量避免.

更重要的是，模型校准缺乏理论基础. EW 模型中的几个因素都是不可交易的，特别是天气变化和断电情况. 因此，在适用于估价的均衡值下，有必要根据天气变化情况和断电概率来调整趋势. 事实上，EW 模型不根据市场价格数据来校准参数，因此完全由关于发电堆的报价堆函数的参数来使模型与市场价格相匹配. 因此，EW 有效假定可观察到的远期价格与边际成本的差异都是由于现货价格可能偏离边际成本，与不可交易因素的非零市场风险价格完全无关.[3] 这是一个强假设，可能对不用于校准模型的工具估价有巨大影响. 例如，这个假设可能会导致对天气或负荷敏感性合约的估价出现很大错误，因为它没有对市场风险成本进行估计.

PJ 提出与之相关但有所不同的方法. 与 EW 模型一样，PJ 模型也是建立在供需基本面基础上的，而不是外生规定电力价格形成过程. 然而 PJ 模型比 EW 模型更加依赖关于负荷和市场报价的现有历史数据. 这些数据越来越广泛存在，并且包含着不应忽略的有价值的信息. 也就是说，PJ 模型利用远期曲线的历史数据来确定发电堆和报价堆的关系，而不是对远期价格曲线进行修正. 这就使得远期价格校准可以集中在确定市场风险价格上. 由于现有的电力衍生品有限，用其来校准报价曲线和市场风险价格是不明智的.

此外，PJ 模型还明确允许主要的不可交易的基础变量，如负荷（或温度），存在非零的市场风险价格，然后根据市场价格校准模型来估计市场风险价格函数. 这是一种很有理论基础的方法，从市场风险价格中提取信息，可以用来对其他一些在修正过程中不被利用的合约进行估价. 而且 PJ 校准方法是以关于问题调整的文献为坚实基础的.

像生活中的所有事情一样，PJ 方法也存在很多权衡. 最重要的是，校准方法要求使用有限次差分法. 由于多元可能带来的问题，这些方法本质上限制了分析中可包含的决定因素的数量. 最初的 PJ 模型仅考虑两个因素：燃料价格和负荷（或温度）. 这些因素明显非常重要. 例如，负荷和燃料价格的波动大约解释了 PJ 模型中高峰期现货价格波动原因的 65%. 尽管如此，从现实世界众多因素中提炼出两个因素重点考虑是明智的. 因为只有一个燃料价格可以很容易地处理，不可能考虑燃料价格和发电堆的整个动态过程. 然而，这个假设没有太大争议，因为在美国天然气正逐渐成为边缘燃料，特别是在很多市场的高峰时段. PJ 还认识到一些其他因素，如断电情况，也会影响电力价格，所以对模型的基本结构进行了适当修改，把这些因素考虑进去了；详见 Pirrong-Jermakyan(2006) 和 Pirrong(2006a，2006b). 不过为了分析简单，本章剩余内容重点讨论 PJ 的基本模型.

8.5 PJ模型概述

负荷是 PJM 模型中电力价格的主要决定变量. 该模型把负荷当做一个可控制的过程. 用 Q_t

表示负荷，$Q_t \leqslant X$，这里 X 表示发电及输电系统的实际能力.[4] 如果负荷超过系统能力，系统会崩溃，给电力用户带来巨大损失. 电力系统管理者（如 PJ 模型中的系统独立营运商）控制负荷并且当负荷达到威胁系统可靠性的水平时，实施干预减少电力用量. Harrison 和 Taksar(1983) 表示，在这些环境下，满足某些技术条件时负荷的可控过程会反映为布朗运动.[5] 为了方便起见，我们对负荷取自然对数，记作 q_t. 对数负荷可以解释这个全微分方程：

$$dq_t = \alpha_q(q_t, t)dt + \sigma_q dW_{qt} - dL_t^u \tag{8.1}$$

其中，L_t^u 是负荷达到临界值的当地时间.[6] 只有当 $q_t = \ln(X)$ 时，L_t^u 是单调递增的（即 $dL_t^u > 0$），否则 $dL_t^u = 0$. 也就是说，lnX 也可以反映 q_t.

源移项 $\alpha_q(q_t, t)$ 与时间 t 有关，这说明负荷不仅随季节变化而且在一天内也会系统地发生变化. 另外，源移项与 q_t 相关允许均值回复. 下面这个方程抓住了这些特性：

$$\alpha_q(q_t, t) = \mu(t) + k[\theta_q(t) - q_t] - 0.5\sigma_q^2 \tag{8.2}$$

其中，σ_q^2 项是通过对数转换产生的. 在这个表达式中，q_t 回归为一个随时间变化的均值 $\theta_q(t)$. $\theta_q(t)$ 可以被指定为一系列正弦值的和，来反映发电量季节性的可预测的波动. 或者它可以当做是一个用非参数经济技术拟合的时间方程. 参数 $k \geqslant 0$ 衡量均值回复的速度；k 值越大，负荷冲击回复越快. 函数 $\mu(t)$ 表示仅依赖时间（尤其是日内时间）的部分负荷变化. 比如，给定 $\theta_q(t) - q_t$，夏季的早上 3 点到下午 5 点负荷趋向于上升，而下午 5 点到早上 3 点趋向于下降.[7]

公式（8.1）的负荷波动率 σ_q 是一个常数，但它随 q_t 和 t 的变化而变化. 关于 σ_q 波动的轻微季节性存在一些经验证明.

第二个状态变量是燃料价格. 边际燃料的远期价格形成过程是：

$$\frac{df_{t,T}}{f_{t,T}} = \alpha_f(f_{t,T}, t) + \sigma_f(f_{t,T}, t)dW_{ft} \tag{8.3}$$

其中，$f_{t,T}$ 是 t 时刻确定的 T 期后交割的远期价格，dW 是标准的布朗运动. 注意 $f_{T,T}$ 是 T 期后的现货燃料价格.

在真实概率测度 $\mathcal{P}$ 下，通过 $\{q_t, f_{t,T}, t \geqslant 0\}$ 这一过程可以解出方程（8.1）和（8.3）. 为了给电力衍生合约订价，我们需要找到一个等价测度 $\mathbb{Q}$，使得在此测度下，取决于 q_t 和 $f_{t,T}$ 的下降了的带有收益的合约价格是鞅值. 由于 $\mathcal{P}$ 和 $\mathbb{Q}$ 一定都有一个 0 测度集，与在 $\mathcal{P}$ 下一样，q_t 在 $\mathbb{Q}$ 下也必须反映 X. 因此，在 $\mathbb{Q}$ 下，q_t 满足下面这个全微分方程：

$$dq_t = [\alpha_q(q_t, t) - \sigma_q\lambda(q_t, t)]tdt + \sigma_q dW_{qt}^* - dL_t^u$$

在这个表达式中，$\lambda(q_t, t)$ 是市场风险价格函数，dW_{qt}^* 是 $\mathbb{Q}$ 鞅值. 由于燃料是一种可贸易资产，在等价测度下，满足 $df_{t,T}/f_{t,T} = \sigma_f dW_{ft}^*$，其中 dW_{qt}^* 是 $\mathbb{Q}$ 鞅值. 源移函数的变化是由于测度的变化导致的.

贴现因子 $Y_t = \exp\left(-\int_0^t r_s ds\right)$，其中 r_s 是 s 时刻的确定利率（后文中假定利率为一个固定常数 r）. 在测度 $\mathbb{Q}$ 下，电力衍生合约的价格发展符合：

$$Y_t C_t = Y_0 C_0 + \int_0^t C_s dY_s + \int_0^t Y_s dC_s$$

在这个表达式中，C_s 表示 s 时刻衍生品的价值，Y_s 表示在 0 时刻确定的 s 期后一美元的价值.

根据 Ito 定理，这个表达式可以重写为：

$$Y_tC_t = C_0 + \int_0^t Y_s\left(\mathcal{A}C + \frac{\partial C}{\partial s} - r_sC_s\right)\mathrm{d}s + \int_0^t\left[\frac{\partial C}{\partial q}\mathrm{d}W_{qs}^* + \frac{\partial C}{\partial f}\mathrm{d}W_{fs}^*\right] - \int_0^t Y_s\frac{\partial C}{\partial q}\mathrm{d}L_s^u$$

其中，$\mathcal{A}$是一个操作量，满足：

$$\mathcal{A}C = \frac{\partial C}{\partial q_t}[\alpha_q(q_t,t) - \sigma_q\lambda(q_t,t)] + 0.5\frac{\partial^2 C}{\partial q_t^2}\sigma_q^2 + 0.5\frac{\partial^2 C}{\partial f_{t,T}^2}\sigma_f^2 f_{t,T}^2 + \frac{\partial^2 C}{\partial q_t \partial f_{t,T}}\sigma_f\sigma_q\rho_{qf}f_{t,T} \tag{8.4}$$

因为电力衍生合约的价格是Q鞅值，所以它必须满足：

$$E\left[\int_0^t Y_s\left(\mathcal{A}C + \frac{\partial C}{\partial s} - r_sC_s\right)\mathrm{d}s\right] = 0$$

和

$$E\left[\int_0^t Y_s\frac{\partial C}{\partial q}\mathrm{d}L_s^u\right] = 0$$

对所有的 t 都成立. 从公式（8.1）可知，$Y_t>0$，并且当 $q_t=X$ 时 $\mathrm{d}L_t^u>0$，利率 r 固定，这些条件可以被重写为：

$$\mathcal{A}C + \frac{\partial C}{\partial t} - rC = 0 \tag{8.5}$$

并且当 $q_t=X$ 时，

$$\frac{\partial C}{\partial q} = 0 \tag{8.6}$$

很明显，式（8.5）和（8.6）足够保证 C 是测度Q下的鞅值；这些条件也是必要的.

式（8.6）是 Neumann 类型的边界条件. 这个边界条件是由于电力市场物理能力的内在限制.[8] 这个限制可以从直觉上来理解. 如果负荷达到最高边界，那么它一定会下降. 如果关于负荷不断变化的衍生品合约在边界是非零的，那么可能存在套利. 例如，如果部分衍生品是负的，卖出合约不会带来损失，而且几乎肯定能带来利润.

式（8.5）可以重写为基本面估价的偏微分方程：

$$rC = \frac{\partial C}{\partial t} + \frac{\partial C}{\partial q_t}[\alpha_q(q_t,t) - \sigma_q\lambda(q_t,t)] + 0.5\frac{\partial^2 C}{\partial q_t^2}\sigma_q^2 + 0.5\frac{\partial^2 C}{\partial f_{t,T}^2}\sigma_f^2 f_{t,T}^2 + \frac{\partial^2 C}{\partial q_t\partial f_{t,T}}\sigma_f\sigma_q\rho_{qf}f_{t,T} \tag{8.7}$$

对于一个远期合约，把时间变量改为 $\tau=T-t$，相关的偏微分方程是：

$$\frac{\partial F_{t,T}}{\partial \tau} = \frac{\partial F_{t,T}}{\partial q_t}[\alpha_q(q_t,t) - \sigma_q\lambda(q_t,t)] + 0.5\frac{\partial_{t,T}^2}{\partial q_t^2}\sigma_q{}^2 + 0.5\frac{\partial_{t,T}^2}{\partial f_{t,T}^2}\sigma_f^2 f_{t,T}^2 + \frac{\partial_{t,T}^2}{\partial q_t\partial f_{t,T}}f_{t,T}\,\sigma_f\sigma_q\rho_{qf} \tag{8.8}$$

其中，$F_{t,T}$是 t 时刻确定的 $T(T>t)$ 期后交割一单位电力的价格.

这是一个二元偏微分方程，其带来的一些问题因利用了发电经济学而在远期合约（而不是期权）情况下可以被大大减轻. 特别地，将电力市场价格看做边际燃料价格和耗热率的产品是很常见的. 耗热率是指生产一单位电力需要的燃料数量. 耗热率是发电单位的边际效率函数，因

为利用不断上升的耗热率单位来满足不断增加的发电量是有效率的，所以这个函数是关于 q 递增的. 这个关系可以在形式上表示为：

$$P_t = f_{t,t}\phi(q_t)$$

其中，P_t 是 t 时刻电力的现货价格，$\phi(q_t)$ 是耗热率函数，$\phi'(q_t)>0$ 且 $\phi''(q_t)>0$. 更一般的形式可以表示成：

$$P_t = f_{t,t}^{\gamma}\phi(q_t)$$

因为在 t 时刻到期的远期合约的价格与那时的现货电力价格有联系，这个函数通过状态变量 f 和 q 来决定远期合约的价格. 而且允许将远期价格的偏微分方程按如下形式重写.

远期价格函数采用下面的形式：

$$F(q_t, f_{t,T}, \tau) = f_{t,T}^{\gamma}V(q_t, \tau)$$

然后将其带入式（8.8），得到新的一元偏微分方程：

$$\frac{\partial V}{\partial \tau} = 0.5\sigma_q^2\frac{\partial^2 V}{\partial q^2} + [\gamma\rho\sigma_f + a]\frac{\partial V}{\partial q} + 0.5\sigma_f^2\gamma(\gamma-1)V \tag{8.9}$$

其中，$a=(\alpha_q(\tau, q)-\sigma_q\lambda(\tau, q))$，解这个方程必须服从 Neuman 边界条件 $\partial V(X, \tau)/\partial q=0$ 和初始条件 $V(q_t, 0)=\phi(q_t)$. 解出这个方程后，就可以通过 $V(\cdot)$ 乘以 $f_{t,T}^{\gamma}$得到远期电力价格. $V(q_t, t, T)$可以理解为 t 时刻的在 T 期后交割的远期市场耗热率. 要解决 $\lambda(q, t)$ 的计算强度很高的逆问题，常用的方法是减少方程变量.

实施这种方法要求知道决定价格的耗热率函数 $\phi(\cdot)$. 对于一些取决于现货价格的合约价格（以远期合约为例），PJ(2006) 提出的方法可以用来把现货价格同状态变量联系起来.[9] 类似地，对于取决于远期价格的合约价格，PJ 模型可以通过将远期价格作为负荷和燃料价格的函数求得远期价格. 这些远期价格又可以反过来求得期权价格与状态变量的关系.

任何情况下，在为电力期权估价前，首先必须求解 PJ 逆问题，来确定市场风险价格函数 $\lambda(q_t)$. 偏微分方程（8.9）的解取决于不能直接观察到的市场风险价格. 但是它可以通过市场上可观察的远期价格得到. 下一节讲述模型校准方法.

8.6 模型校准

一开始似乎没有必要估计负荷波动风险的市场价格. 如果我们认为只需要知道系统性风险的均衡价格，特殊风险不需要定价，因为电力价格几乎与整个市场表现的测量值无关，那么我们可能会认为 λ 应该为 0. 这是错误的. PJ(2006) 模型中存在广泛证据证明在电力市场风险价格很重要. Bessembinder 和 Lemon 在 2002 年的研究中说明，这反映了电力市场与金融系统的不完全融合.

因为风险的市场价格潜力很大，在为衍生品估价时需要被考虑. 遗憾的是，风险的市场价格不能被直接观测到. 然而可以从交易工具的价格推断出风险价格. 特别是当给定一系列报出的远期价格时，可以对式（8.9）应用逆问题法求得包含的风险市场价格. 这些方法本质上是找到一个 λ 函数，根据市场报出的价格校准模型产生的远期价格. 这与根据观测到的零息债券价格来校准即期利率模型（如 Hull-White 模型）很类似.

在校准中很重要的事情是不要过度拟合（在利率模型中就经常出现这种情况）. 例如，如果

有 12 个远期价格需要校准，选择一个正好拟合这个价格的 12 阶多项式的 λ 函数总是可能的. 这具有欺骗性，并且校准结果将会不稳定. 用来校准模型的价格发生非常轻微的波动，比如说从中间价变成买入价或卖出价，都将会导致估计的 λ 函数发生非常大的变化. 类似地，过度拟合校准每天都发生很大变化. 这种问题被称作"不利姿态". 要解决这些问题必须对函数加上一些限制来惩罚过度拟合. 这被称作"正则化". [10]

为形式地来看这个问题，需要注意在任何时候市场报出的远期价格都是有限的. 将现有的远期报价集合记作$\mathcal{D}$. 正则化技术涉及选择一个 λ 函数，在服从正则化限制的前提下，使得一系列给定的交割期通过式（8.9）得到的远期价格与市场报出的价格之间的偏差平方和最小. 为了使问题更容易掌控，λ 仅仅是负荷的函数. PJ 模型用 H^2 范数（norm）来正则化：

$$R(\lambda) = \int_{\ln(\hat{X})}^{\ln(X)} \left[\lambda^2 + \left(\frac{\partial \lambda}{\partial q}\right)^2\right] dq$$

其中，X 是最大负荷（由发电系统的物理能力决定），$\ln(\hat{f})$ 是最小负荷水平（如果负荷太小，将会导致发电系统不稳定）. 然后选择一个 λ 函数，使下列式子最小化：

$$\sum_{i \in \mathcal{D}} [F_i(q_t, f_{t,T} \mid \lambda) - \mathcal{F}_i]^2 + kR(\lambda)$$

在这个式子里，$F_i(q_t,\ f_{t,T} \mid \lambda)$ 是给定的当前负荷 q_t 和燃料价格 $f_{t,T}$对应 $i \in \mathcal{D}$的报出的远期价格通过式（8.8）求得的远期合约价格；注意，远期价格依赖于 λ 函数. $\mathcal{F}_i$ 表示第 i 个远期报价. k 为正则化参数.

正则化技术从根本上惩罚了过度拟合. 在正则化问题中，存在一个在远期报价拟合精确度和 λ 函数平滑性之间的权衡. 注意，当 $|\partial\lambda/\partial q|$ 很大时，$R(\lambda)$ 很大. 因此，当 λ 非常不平滑（平滑）时，$R(\lambda)$ 值很大（小）. 正则化参数 k 的选择决定了拟合结果的平滑性；这个参数值越大，对非平滑性的惩罚越大，结果就越平滑.

用有限次差分法来解决这个问题. 用 q 和 τ 的增量 Δq 和 $\Delta\tau$ 来建立一个估价网格；总共有 N 个时间阶段和 M 个 q 值. 对 λ 有一个初始猜测. 在估价网格里，λ 被表示为一个向量，向量的长度等于 M.

用小参数方法继续进行估计. 具体地，用 λ_0 来表示对 λ 函数的初始猜测，将 λ 表示为：

$$\lambda(q) = \lambda_0(q) + \sum_{j=1}^{\infty} \lambda_k(q)$$

其中，λ_k 是对初始猜测的改进. 类似地，V_0 表示式（8.13）基于 λ_0 的解，并令：

$$V = V_0 + \sum_{j=1}^{\infty} V_k$$

其中，V_k 是对初始猜测的改进. [11]注意，$V(\cdot)$ 函数利用了更少的变量.

考虑一个市场风险价格的 ε 型函数：

$$\lambda_{(q)} = \lambda_0 + \sum_{j=1}^{\infty} \varepsilon^k \lambda_k(q)$$

对应的 V 的表达式为：

$$V(\tau, q) = V_0(\tau, q) + \sum_{j=1}^{\infty} \varepsilon^k V_k(\tau, q)$$

将这些式子带入估价的偏微分方程并用 ε 来表示电力，得到：

$$\frac{\partial V_0}{\partial \tau} = c_0 \frac{\partial^2 V_0}{\partial q^2} + c_1[\lambda_0] \frac{\partial V_0}{\partial q} + c_0 V_0 \tag{8.10}$$

$$\frac{\partial V_1}{\partial \tau} = c_0 \frac{\partial^2 V_1}{\partial q^2} + c_1[\lambda_0] \frac{\partial V_1}{\partial q} + c_0 V_1 - \sigma_q q \lambda_1 \frac{\partial V_0}{\partial q} \tag{8.11}$$

$$\frac{\partial V_k}{\partial \tau} = c_0 \frac{\partial^2 V_k}{\partial q^2} + c_1[\lambda_0] \frac{\partial V_k}{\partial q} + c_0 V_k - \sum_{j=0}^{k} \sigma_q q \lambda_j \frac{\partial V_{k-j}}{\partial q} \tag{8.12}$$

这里 $k=2$，3，…其中，$c_0=0.5\sigma_f^2\gamma(\gamma-1)$，$c_1[\lambda_0]=q[\gamma\rho\sigma_q\sigma_f+\alpha_q(\tau, q)-\sigma_q\lambda_0(\tau, q)]$，$c_2=0.5\sigma_q^2q^2$. 在分析中针对每一到期日有一系列方程（8.10）～（8.12）.

偏微分方程（8.10）可以根据暗示条件解出. 从式（8.11）可以看出，离散的 q 和 τ 之间存在递归关系：

$$A_{n+1}V_1^{(i)}[n+1] = V_1^{(i)}[n] + G_{n+1}^{(i)}\lambda_1 \tag{8.13}$$

其中，$n+1$ 表示时间点，A_{n+1} 是一个由偏微分方程的内在定理和系数决定的 $(M-2)\times(M-2)$的三角矩阵，$V_1^{(i)}[n]$ 是一个 $(M-2)$ 维向量，表示时间点 n 对到期日 $i\in\mathcal{D}$每一个内部负荷的 V_1 值，并且

$$G_{n+1}^{(i)} = -0.5\frac{\Delta\tau}{\Delta q}\sigma_q q \frac{\partial V_0^{(i)}(q,(n+1)\Delta\tau)}{\partial q}$$

其中，偏导数是用集中有限差分来估计的. 而且当 $n\Delta t$ 大于远期合约 $i\in\mathcal{D}$距到期日的时间时，$G_n^{(i)}=0$. 完成这个递归过程可得：

$$V_1^{(i)}[N] = [\sum_{j=1}^{N-1}(\prod_{k=0}^{j}A_{N-j})G_{N-j}^{(i)}]\lambda_1 = \mathcal{B}^{(i)}\lambda_1 \tag{8.14}$$

注意，正则化后的目标函数变成：

$$\sum_{i\in\mathcal{D}}\left[V_0^{(i)} + \mathcal{B}^{(i)}\lambda_1 - \frac{\mathcal{F}_i}{f_i^{\gamma}}\right]^2 + kR(\lambda_1) \tag{8.15}$$

其中，f_i 是与电力远期合约 i 的到期日相同的燃料远期价格. 用梯形规则来近似正则化公式中的积分后，$R(\lambda_1)$ 是关于 λ_1 的二次方程. 因此，使式（8.14）关于 λ_1 最小化可得到一系列线性方程（一阶条件），从而解出 λ_1.

鉴于此改进，基于式（8.12），一种类似的方法可用来解 $\lambda_k(k>1)$；$k=2$，3，…与 $k=1$ 时的解法的主要区别在于额外 forcing 项的存在，这些限制项取决于 V 改进$k-1$，$k-2$，…，2 的 q 次微分. 可以自行选择实施改进的总次数，次数越多，计算成本（尤其是存储成本）越高.

图 8.2 描绘了基于 4 个改进拟合 7 个远期价格的 λ 函数图像，这 7 个远期价格是 PJ 模型在 2005 年 6 月 5 号（2005 年 6 月至 12 月交割）观测到的. 由图 8.2 可知，风险的市场价格函数值总是负的. 这意味着负荷在等价测度下比在物理测度下上升得更加迅速. 这使得负荷密度向右倾斜. 由于电力价格关于负荷单调递增，这反过来暗示远期价格被高估了；也就是说，等价测度下的期望价格高于物理测度下的期望价格. 还要注意 λ 函数的绝对值是关于负荷递增的，这意味着对于在高需求时期到期的远期合约，其高估幅度更大.

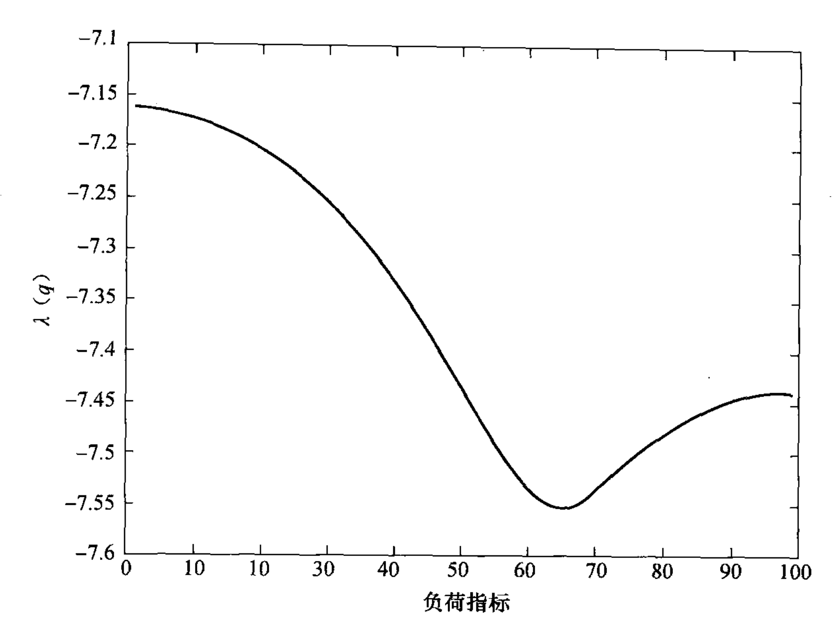


图 8.2 PJ 模型的风险市场价格函数

8.7 利用校准模型进行期权定价

经拟合得到 λ 函数后，利用模型给在校准过程中没被利用的期权定价是可能的. 存在大量电力期权交易（主要是场外交易). 最常见的是日执行期权、月执行期权和点火价差期权. 依次考虑每一种期权.

8.7.1 日执行期权

日执行期权（daily strike option）的价格依赖于每一天的电力价格，特别是一天中高峰时段的电力交割价格.

日执行期权可以是非现金交割或现金交割. 对于一个非现金交割的日执行看涨期权，行权时期权所有者可获得一份每日交割远期合约的多头头寸，被授予在当天高峰时段交割一定量电力的权利. 看跌期权的多头行权时可得每日交割远期合约的空头头寸. 期权所有者必须在交割期前（比如，交割日的前一天）决定是否行权.

现金交割的日执行期权可以用多种方式来建立. 例如，现金交割的日执行看涨期权的多头可获得一笔金额，由 0 和到期日相同的相关远期价格与期权执行价的差价两者较高者决定. 举个例子，一个期权多头星期二的价值可能依赖于在星期三交割的星期二的远期价格. 或者日执行看涨期权的多头也有可能要支付定价日观测到的平均现货价格与执行价格之间的差价. 比如说，看涨期权多头要支付一笔金额，由 0 和周三的平均现货价格与执行价格的差价中较高者决定. 在集中实时市场（如 PJM）构造有这种支付结构的期权尤其可行.

我关键看价值依赖于远期价格的日执行期权.[12] 对于这种期权，以 T 期后到期的远期合约为标的资产的 t' 期后到期的看涨期权的价值是 $(F_{t',T}(q_{t'},\ f_{t',T})\ -K)^{+}$，看跌期权的价值是

$(K-F_{t',T}(q_{t'}, f_{t',T}))^+$.

8.7.2 月执行期权

月执行看涨期权的多头行权时可获得一份每月交割远期合约的多头头寸. 比如，7 月份交割的看涨期权多头在 6 月末行权时可获得一份远期合约，有权在即将到来的 7 月的高峰时段交割一定量的电力. 用 $F_{t',j}$表示期权行权日为t'，在期权合约月份的第 j 天进行电力交割的远期价格，月执行看涨期权的价值是：

$$\left(\frac{\sum_{j\in M}F_{t',j}}{\sum_{j\in M}\delta_j}-K\right)^+$$

其中，M 是合约月中的一系列交割日，δ_j 是一个指示变量，当 $j\in M$ 时取值为 1，否则为 0.

8.7.3 点火价差期权

点火价差看涨期权的价值等于 0 和远期价格与燃料价格差价两者之间的较大值乘以合约规定的耗热率. 耗热率用每百万英国热量单位（mmBTU）的兆瓦数（MW）来衡量. 电力生产的边际成本等于耗热率乘以燃料价格. 因此，点火价差期权可以被看做是一种使用燃料发电的期权，因为期权价值是以电力价格与耗热率一定的发电成本之间的差价为基础的. 从这点看来，发电厂常被视作点火价差期权的集合体，尽管点火价差期权是作为独立的金融产品进行交易.

点火价差期权同日执行期权一样，也存在一些关于行权时间、非现金交割和现金交割的问题. 如果点火价差期权必须在电力交割期 T 之前的某个时间 t'行权，那么看涨期权的收益是 $(F_{t',T}-f_{t',T}H^*)^+$，它有效地决定是否行权，H^*是合约规定的耗热率.

8.8 期权估价方法

8.8.1 分裂（有限次）差分：日执行和月执行期权

可以通过用有限次差分法解偏微分方程（8.7）来对日执行和月执行期权进行估值. 这种方法是由 Duffy 在 2006 年描述的，可以处理非零相关并且计算上是有效的. 处理非零相关的能力使得这种方法比交替方向隐式技术更受欢迎. [13] 裂分法也允许运用自然临界限制——最重要的是对负荷的 Neumann 限制和对燃料的传统 Dirichlet 限制.

因为这是一种有限次差分法，首先要建立一个关于时间、对数燃料价格和对数负荷的网格；燃料价格的对数转换可以保证有一个裂分方程含有固定系数. 时间增量记作 δt；考虑到负荷的季节性，为了方便起见，$\delta t=1/365$. 对数燃料价格的增量记作 $\delta\hat{f}$，对数负荷的增量记作 δq.

顾名思义，裂分法就是在每个时间点把偏微分方程（8.7）分裂成两部分. 经对数转换后，第一部分是：

$$rC=\frac{\partial C}{\partial t}+\frac{\partial C}{\partial q_t}[\alpha_q(q_t,t)-\sigma_q\lambda(q_t,t)]+0.5\frac{\partial^2 C}{\partial q_t^2}\sigma_q^2+0.5\sigma_f\sigma_q\rho_{qf}\frac{\partial^2 C}{\partial q_t\partial\hat{f}_{t,T}} \tag{8.16}$$

第二部分是：

$$0=\frac{\partial C}{\partial t}-0.5\sigma_f^2\frac{\partial C}{\partial \hat{f}}+0.5\sigma_f^2\frac{\partial^2 C}{\partial \hat{f}_{t,T}^2}+0.5\sigma_f\sigma_q\rho_{qf}+\frac{\partial^2 C}{\partial q_t\partial \hat{f}_{t,T}} \tag{8.17}$$

在到期日之前的一个时间点，用隐式法解方程（8.16），得到从第二低到第二高的每一个不同的燃料价格，在 Yanenko(1971）之后，利用外部近似值（根据这个时间点之后的 C 值）来设定交叉偏导数. 对于到期日之前的一个时间点，期权价值被当作初始条件. 由方程(8.17)给出的先前时间点的期权价值被用于更早时期的期权估价. 解方程时会用到对每一个燃料价格的 Neumann 边界限制. 对于最高和最低的燃料价格，运用 Dirichlet 边界限制. 比如说，对一个看跌期权，当燃料价格最高时，看跌期权价值为 0，而当燃料价格最低时，看跌期权的价格等于执行价与电力远期价格之间的差价乘以期权到期时支付的 1 美元的现值. 由于均值回复速度很快，方程（8.16）的一阶条件项的系数绝对值通常很大. 因此尽管式（8.7）是一个对流扩散方程，但是其对流效应比金融领域常见的抛物线式的偏微分方程更加重要，要解决对流问题，离散化方法比较常用. 特别地，当系数为负时，用前向差分来估计 $\partial C/\partial q$；系数为正时，后向差分.

在每个时间点，式（8.16）的先前时间点的解作为式（8.17）的初始条件，然后求得式(8.17）的解，从第二低到第二高的每一个对数负荷. 所有这些解都运用 Dirichlet 边界限制. 对于最低和最高的对数负荷，运用 Neumann 边界限制.

这种方法在计算上相对有效. 用 Matlab 为这个模型编码时，利用每天的时间点，在一个 2.4GHz 的奔腾计算机上估计一个月到期的期权价值只需要花大约 2 秒的时间.

日执行期权的价值是这样决定的. 假定期权多头必须在电力交割日之前决定行权，即 $t'=T-\delta t$.[14] 行权时，对于已知的负荷 q 和对数燃料价格 $\hat{f}$，看涨期权的多头可以得到一笔报酬，等于 0 和前一天远期价格 $F_{t',T}(q, f)$ 与执行价的差价两者之间的较高值，远期价格 $F_{t',T}(q, f)$ 是经观察到的价格曲线校准后的 Pirrong-Jermakyan 模型的解.[15]

对于月执行期权，合约月份的交割日首先被确定了. 为了简单起见，假定交割都发生在合约月份的每一工作日的高峰时段. 期权要在第一个交易日之前的工作日执行. 在这一天，Pirrong-Jermakyan 模型中交割月的每一天的远期价格都已经根据网格中 $\hat{f}$ 和 q 的值确定下来.[16] 例如，在 6 月 30 号，可以确定在 7 月 1 号至 7 月 31 号的所有工作日到期的远期价格. 看涨期权的执行结果等于 0 和这些远期价格平均值与执行价的差价两者之间的较高值.

8.8.2 月执行期权估价的 Matlab 应用

用 Matlab 编码利用裂分法给月执行期权估价，可以略微修改程序来为日执行期权估价. 程序的几个步骤如下：

- 程序把负荷波动率 `sigmaq`、燃料波动率 `sigmaf`、均值回复速度 `mrcoef` 和漂移系数矩阵 `new_qdrift` 当做输入变量. 波动率和均值回复是标量. `new_qdrift` 是一个 $nq\times nt$ 矩阵，其中 nq 是负荷不同值的数量，nt 是时间点的数量. 由于均值回复和风险市场价格与负荷有关并且对数负荷均值随时间变化，负荷漂移项 $\alpha(\cdot)$ 随负荷水平和时间变化，所以在每一个对数负荷-时间点上，都存在不同的漂移项. `new_qdrift` 可以用 Pirrong-Jermakyan(2006）描述的手段和之前描述的校准方法来确定.
- `new_qdrift` 用来构造 `c1(:,:)`，式（8.7）中 q 的一阶项的 $nq\times nt$ 系数矩阵. 另外，

sigmaq 用来构造 c2(:,:)，式（8.7）中 q 的二阶项的 $nq \times nt$ 系数矩阵；利率 r 用来构造 c0(:, 1)，0 阶项的系数矩阵.

- 程序利用一个 4×1 向量，option_date_vec，来确定与价值相关的日期和期权的到期日. 这个向量被用来构造一个由 0 和 1 构成的 $nt \times 1$ 向量，1 对应对期权价值有贡献的日期. 例如，对一份 7 月到期的高峰电力的月执行期权，向量 monthly_strike_vec 在对应 7 月份每一天的行中值为 1. 向量 monthly_strike_vec 除以 sum(monthly_strike_vec) 得到向量 monthly_strike_dummy. 这被用来获得包含在月执行期权交割集中的平均远期价格，以确定期权价值和燃料价格的上下限.
- 程序的核心是在裂分中运用的有限次差分法. 程序时间跨度从期权到期日到现在日期. 在每一个时间点，方程（8.16）可以按下面方法对负的 c1 离散化：

$$\frac{C_{i,j}^{k+0.5}-C_{i,j}^{k}}{\delta t}=c1_{i,k+1}\frac{C_{i,j}^{k+0.5}-C_{i-1,j}^{k+0.5}}{2\delta q}+c2_{i,k+1}\frac{C_{i+1,j}^{k+0.5}-2C_{i,j}^{k+0.5}+C_{i-1,j}^{k+0.5}}{\delta q^{2}}$$
$$+0.5\rho\sigma_{q}\sigma_{f}\frac{C_{i+1,j+1}^{k}-C_{i+1,j-1}^{k}-C_{i-1,j+1}^{k}+C_{i-1,j-1}^{k}}{4\delta q\delta\hat{f}}-rC_{i,j}^{k+1}$$

其中，上标表示到期前衡量期权价格 C 的时间点的数量，第一个下标 $2 \leqslant i \leqslant nq-1$ 表示负荷网格中的点，第二个下标 $2 \leqslant j \leqslant nf-2$ 表示燃料网格中的点.[17] 注意，系数与时间和负荷都相关. 还要注意这个表达式允许根据时间点 k 的期权价值来确定在 $k+0.5$ 这样的虚时间点上的期权价值. 将每个时间点上的项汇集起来可得到一个 nq- 2xnq- 2xnt- 1 的数组 mlarray，使得 mlarray(:,:, k+ 1) * u_array(2: nq- 2, k+ 0.5)= u_array(2: nq- 2, k)+ corr_correction，其中，u_array(:, k+ 0.5) 是虚拟时间点 $k+0.5$ 上的期权价值，corr_correction 是这个价值与 ρ 的乘积，取决于时间点 k 上的 u_array 值. 可以通过转置矩阵或者 Matlab 编码解出这一系列方程，确定虚拟时间点 $k+0.5$ 上的内部对数负荷对应的期权价值. 应用 Neumann 边界条件可以确定最高和最低对数负荷的期权价值. 这一系列方程是对应每一个燃料价格点 j= 2, …, nf- 1 求解的，这些价格点都在一个燃料价格圈里（j1 圈）. 对于 j= 1 和 j= nf，可以运用 Dirichlet 边界条件来确定每一负荷对应的期权价值. 虚拟时间点 $k+0.5$ 的 u_array 值一经确定，经对数转换后，通过如下对式（8.17）的离散化，程序可以进行到完全时间点 $k+1$：

$$\frac{C_{i,j}^{k+1}-C_{i,j}^{k+0.5}}{\delta t}=-0.5\sigma_{f}^{2}\frac{C_{i,j+1}^{k+1}-C_{i,j-1}^{k+1}}{2\delta\hat{f}}+0.5\sigma_{f}^{2}\frac{C_{i,j+1}^{k+1}-2C_{i,j}^{k+1}+C_{i,j-1}^{k+1}}{\delta\hat{f}^{2}}$$
$$+0.5\rho\sigma_{q}\sigma_{f}\frac{C_{i+1,j+1}^{k+0.5}-C_{i+1,j-1}^{k+0.5}-C_{i-1,j+1}^{k+0.5}-C_{i-1,j-1}^{k+0.5}}{4\delta q\delta\hat{f}}$$

把公共项再次汇集可以得到一系列线性方程，解得完全时间点上的 u_array 值. 将这列线性方程的矩阵记为 mlarray_f. 通过这列方程可以解得每一个负荷 i= 2, …, nq- 1，方程解是在一个负荷圈内（j2 圈）. 对每一个不同的负荷，Dirichlet 边界条件的运用可以确定最高和最低燃料价格的 u_array 值. 对于 i= 1 和 i= nq，以及对应的 j= 1, …, nf，运用 Neumann 边界条件来求期权价值. 以上所述都是求看跌期权的价值. 然后可以用看跌看涨平价来求看涨期权的价值.

下面是月执行期权估价的 Matlab 编码：

power_option_splitting_london.m

```
function v=power_option_splitting(vall,strike,date_vec,sigmaq,sigmaf,rho,
  qvec,new_qdrift,cptype,option_date_vec,nq,nf,r)

% THIS PROGRAM USES A SPLITTING TECHNIQUE TO SOLVE THE 2D PDE THAT VALUES A
% MONTHLY STRIKE OPTION.  THE TECHNIQUE CAN BE READILY ADAPTED TO SOLVE FOR
% THE VALUE OF A DAILY STRIKE OPTION

% THE FUNCTION TAKES INPUTS:

% nq:=number of log load points in the valuation grid
% nf:=number of log fuel points in the valuation grid
% vall:=a matrix of forward market heat rates for each date between the
%   present and option expiry.  vall is determined by solving the
%   Pirrong-Jermakyan calibration problem applied to observed power forward
%   prices and fuel prices
% qvec:=nq x 1 vector of log load points.  Each row in vall corresponds to
%   a log load point in qvec
% strike:=the monthly option strike in $/MWh
% sigmaq:=load volatility determined from historical data--see
%   Pirrong-Jermakyan (2006)
% sigmaf:=fuel volatility
% rho:=load-fuel correlation
% new_qdrift:=matrix of load drifts (in the equivalent measure)
% cptype:= option type indicator -1 for a put +1 for a call
% option_date_vec:=a 4x1 vector of dates, with the first element the
% current date, the second element the option expiration date, the third
% element the first forward delivery date included in the monthly strike
% bundle, and the last (fourth) element the last forward delivery date
% included in the bundle

ndates=size(vall,3);
nt=ndates;
nm=size(vall,2);

sigmaq2=sigmaq^2;


% determine 1st, 2nd, and 0 order load coefficients in valuation PDE

for jtc=1:nt

    c1(:,jtc)=new_qdrift(:,jtc);
    c2(:,jtc)=.5*sigmaq2*ones(nq,1);
    c0(:,jtc)=r*ones(nq,1);

end

% create al, bl, cl, and ar, br, and cr matrices

% assume daily time steps

dt=1/365;

nu1=dt/(dq);
nu2=dt/((dq^2));

ar=zeros(nq,nt-1);
br=ar;
cr=ar;
```

```
al=ar;
bl=ar;
cl=ar;

% centdiff is an indicator that equals zero when using upwind/downwind
% differencing (depending on the sign of c1) on the 1st order load term and
% equals 1 when using central differencing.  Given the magnitude of c1,
% non-central differencing typically preferable

centdiff=0;

for jta=1:nt-1

    if centdiff==0

        c1dum=(c1(:,jta)>0);
        al(:,jta)=-nu2*c2(:,jta)+nu1*(1-c1dum).*c1(:,jta);
        bl(:,jta)=1-dt*c0(:,jta)+2*nu2*c2(:,jta)+nu1*(c1dum.*c1(:,jta)-
          (1-c1dum).*c1(:,jta));

        % use cx instead of cl (c letter l) to avoid confusion with c1
        % (c numeral one)

        cx(:,jta)=-nu2*c2(:,jta)-nu1*c1dum.*c1(:,jta);

    else


    % central differences

        al(:,jta)=-nu2*c2(:,jta)+.5*nu1*c1(:,jta);
        bl(:,jta)=1-dt*c0(:,jta)+2*c2(:,jta)*nu2;
        cx(:,jta)=-nu2*c2(:,jta)-.5*c1(:,jta)*nu1;

    end

end

% fill in mlarray and mlarrayinv
% mlarray is a matrix implied by the discretization of the first (load)
% split in the valuation PDE
% this array is used to determine option values at pseudo-step

mlarray=zeros(nq-2,nq-2,nt-1);
mrarray=zeros(nq-2,nq,nt-1);

for jtl=1:nt-1
    for jqr=2:nq-3
        for jqc=1:nq-2;
            if jqc==jqr
                mlarray(jqr,jqc,jtl)=bl(jqr+1,jtl);
            end

            if jqc==jqr+1
                mlarray(jqr,jqc,jtl)=cx(jqr+1,jtl);
            end

            if jqc==jqr-1
                mlarray(jqr,jqc,jtl)=al(jqr+1,jtl);
```

```
            end

        end
    end

% fill in values of mlarray implied by load boundary conditions
% use von Neumann boundary condition implied by reflecting
% barriers at upper and lower load levels

    mlarray(1,1,jt1)=al(2,jt1)+(4/3)*bl(2,jt1);
    mlarray(1,2,jt1)=cx(2,jt1)-(1/3)*bl(2,jt1);
    mlarray(nq-2,nq-2,jt1)=(4/3)*bl(nq-1,jt1)+cx(nq-1,jt1);
    mlarray(nq-2,nq-3,jt1)=al(nq-1,jt1)-(1/3)*bl(nq-1,jt1);


end

% Define fuel grid
% use fuel prices that are well away from current fuel prices

    F_min=2;
    F_max=16;

    f_min=log(F_min);
    f_max=log(F_max);

    df=(f_max-f_min)/(nf-1);
    f=f_min:df:f_max;


% now define terms that will determine mlarray_f
% this array is used to solve for option values at the full step

A_f=-.5*(sigmaf^2)*((dt/df^2)+.5*(dt/df));
B_f=1+(sigmaf^2)*(dt/df^2);
C_f=.5*(sigmaf^2)*(-(dt/df^2)+.5*(dt/df));

% determine mlarray_f

    for jfr=1:nf-2
        for jfc=1:nf-2;

            if jfc==jfr
                mlarray_f(jfr,jfc)=B_f;
            end
            if jfc==jfr+1
                mlarray_f(jfr,jfc)=C_f;
            end

            if jfc==jfr-1
                mlarray_f(jfr,jfc)=A_f;
            end

        end
    end
% payoff matrix
% it is assumed that monthly strike option requires delivery of a bundle of
% on peak forward price with delivery during weekdays
% therefore, need a weekday indicator variable wday taking values of one
```

```
% during weekdays and zero otherwise

[wday,w]=weekday(date_vec);

% cdate:=current date
% edate:=option expiry
% fdeldate:=first delivery date in bundle
% ldeldate:=last delivery date in bundle

    cdate=option_date_vec(1);
    edate=option_date_vec(2);
    fdeldate=option_date_vec(3);
    ldeldate=option_date_vec(4);

% monthly_strike_vec:=zeros for non-delivery dates, ones for delivery dates

    monthly_strike_vec=(date_vec>=fdeldate).*(date_vec<=ldeldate).*
      (wday>1).*(wday<7);

% monthly_strike_dummy:=set of weights to be applied to each deliverable in
%   bundle to determine average price

    monthly_strike_dummy=monthly_strike_vec(2:nt)/
      sum(monthly_strike_vec(2:nt));

% jexpiry is the number of days until expiration

    jexpiry=edate-cdate+1;

% determine option payoff.  Multiply heat rate array for expiration date by
% monthly_strike_dummy to determine average heat rate.  Second dimension of
% vall is a delivery date dimension

    vpaymat=vall(:,:,jexpiry)*monthly_strike_dummy;

% rvec will be used to implement Dirichlet boundary conditions (in
% fuel dimension)
    rvec=zeros(nf-2,1);

% cross correlation term

    cross_coefficient=rho*sigmaf*sigmaq*dt/(dq*df);


% can do payoff as put or a call
% or can implement by doing payoff for put only, then using put call
% parity to value call
% desirable to use a put because don't have to worry about value as f
% goes to infinity or 0
% payoff for a put is strike minus average market heat rate for monthly
% strike bundle times fuel price for that bundle

    for jf=1:nf
        payoff(:,jf)=max(-vpaymat*exp(f(jf))+strike,0);

    end


% initialize u_array
```

```
% u_array will contain the option values for each date from the present
% to expiry
% since option value is solved for at whole and half (pseudo) time steps,
% u_array consists of 2*jexpiry-1 nq x nf matrices

    u_array=zeros(nq,nf,2*jexpiry-1);

% u_array value at expiry given by option payoff

    u_array(:,:,2*jexpiry-1)=payoff;

% begin time stepping from expiry to the present

    for jt=jexpiry-1:-1:1


% Implementation of splitting methodology
% first loop over fuel steps
% this solves at the half step in q implicitly

        for j1=2:nf-1

% corr_correction term arises from cross derivative in PDE
% use explicit approximation per Yanenko

            corr_correction=(u_array(3:nq,j1+1,2*jt+1)-
              u_array(3:nq,j1-1,2*jt+1));
            corr_correction=corr_correction+(-u_array(1:nq-2,j1+1,2*jt+1)+
              u_array(1:nq-2,j1-1,2*jt+1));
            corr_correction=.125*cross_coefficient*corr_correction;

            u_array(2:nq-1,j1,2*jt)=mlarray(:,:,jt)\
              (u_array(2:nq-1,j1,2*jt+1)+corr_correction);

% von Neumann boundary conditions

            u_array(1,j1,2*jt)=(4/3)*u_array(2,j1,2*jt)-
              (1/3)*u_array(3,j1,2*jt);
            u_array(nq,j1,2*jt)=(4/3)*u_array(nq-1,j1,2*jt)-
              (1/3)*u_array(nq-2,j1,2*jt);

        end % end j1 loop

% fill in fuel boundaries--use Dirichlet BC

            v_bound=vall(:,:,jt)*monthly_strike_dummy;
            u_array(:,1,2*jt)=exp(-r*dt*(jexpiry-jt))*(-exp(f(1))*
              v_bound+strike);;
            u_array(:,nf,2*jt)=0;


% now do the fuel split so loop over load steps

        for j2=2:nq-1

            rvec(nf-2,1)=0;
            rvec(1,1)=-A_f*exp(-r*dt*(jexpiry-jt))*(-exp(f(1))*
              vall(j2,:,jt)*monthly_strike_dummy+strike);
            corr_correction=(u_array(j2+1,3:nf,2*jt)-
              u array(j2+1,1:nf-2,2*jt));
```

```
            corr_correction=corr_correction+(-u_array(j2-1,3:nf,2*jt)+
              u_array(j2-1,1:nf-2,2*jt));
            corr_correction=.125*cross_coefficient*corr_correction';
            rhs_vector=u_array(j2,2:nf-1,2*jt)'+corr_correction+rvec;
            u_array(j2,2:nf-1,2*jt-1)=(mlarray_f\rhs_vector)';

% Dirichlet boundary conditions

            u_array(j2,1,2*jt-1)=-exp(-r*dt*(jexpiry-jt))*(exp(f(1))*
              vall(j2,:,jt)*monthly_strike_dummy-strike);
            u_array(j2,nf,2*jt-1)=0;


        end % end j2 loop

% fill in values for lowest and highest loads
% use von Neumann boundary conditions


        u_array(1,:,2*jt-1)=(4/3)*u_array(2,:,2*jt-1)-
          (1/3)*u_array(3,:,2*jt-1);
        u_array(nq,:,2*jt-1)=(4/3)*u_array(nq-1,:,2*jt-1)-
          (1/3)*u_array(nq-2,:,2*jt-1);


    end % end jt loop

    v=zeros(nq,nf,jexpiry);

    clear u_array

% convert v if option is a put using put call parity
% only use full step values

    if cptype>0

        for jt=1:jexpiry
            for j2=1:nf


                fvec=exp(f(j2))*vall(:,:,jt)*monthly_strike_dummy;
                v(:,j2,jt)=u_array(:,j2,2*jt-1)+
                  exp(-r*dt*(jexpiry-jt))*(fvec-strike);


            end
        end

    else

       for jt=1:jexpiry

            v(:,:,jt)=u_array(:,j2,2*jt-1);


        end

    end % end cptype>0

% Now, that was easy, wasn't it?
```

8.8.3 点火价差期权

因为在 PJ 模型中，远期价格是一个可分函数，可表示成燃料远期价格与负荷函数的乘积，所以点火价差期权的价值可以重新表示为：

$$(F_{t',T}-f_{t',T}H^*)^+=(f_{t',T}V(q_{t'},t'T)-f_{t',T}H^*)^+=f_{t'T}(V(q_t,t',T)-H^*)^+$$

因此，点火价差期权的价值可分为负荷与燃料的乘积. 所以运用之前讨论过的 PJ 分解法将点火价差期权价值写成另一个可分为当前燃料远期价格与当前负荷乘积的函数是可能的. 把点火价差期权价值记作 $H(\cdot)$：

$$H(q_t,f_{t,T},t,T,H^*)=f_{t,T}\Phi(q_t,t,T,H^*)$$

$\Phi(\cdot)$ 函数可以用一个以 $(V(q_t,t'T)-H^*)^+$ 作为初始条件的标准内在求解程序来确定.[18]

8.8.4 点火价差期权定价的 Matlab 应用

由于维度减少了，点火价差期权的编程相对简单. 因此，我们只需要构建一个关于负荷和时间的网格. Matlab 编码确定对应每一个负荷、每一个到期日的看涨点火价差期权的价值. 然后将价值传入方程 fpowdirect_implicit_sparkspread_1.m. 这个方程也得到一个负荷漂移项的系数矩阵 drift_mat，该矩阵反映了 $\alpha(\cdot)$ 和 $\lambda(\cdot)$ 函数，负荷与燃料波动率以及与时间和负荷网格相关的信息. 该方程用一个内在的偏微分方程（单边微分更优）求解程序来确定对应时间和负荷的每一个到期日的 $\Phi(\cdot)$ 函数. 方程结果用每兆瓦小时（MWh）的 MMBTU 来衡量. 为了把这个值转换成美元/兆瓦小时，可以将方程结果与适当的燃料远期价格相乘. 例如，对于 2007 年 7 月到期的点火价差期权，将方程结果与 2007 年 7 月的燃料远期价格相乘.

以下是点火价差期权估价的 Matlab 编码. 这个 DOS 命令需要一个内在的偏微分方程求解程序 fpowdirect_implict_sparkspread_1.

Spark_speed_shell.m

```
% spark spread option code uses dimensionality reduction and 1D
% implicit solver assuming pgbeta (gamma)==1.
% The solver determines the value of spark spread option in a portfolio
% of said options, the first of which expires on fdeldate and the last
% on ldeldate

% date manipulations

cdate=option_date_vec(1);
edate=option_date_vec(2);
fdeldate=option_date_vec(3);
ldeldate=option_date_vec(4);

jexpiry0=fdeldate-cdate+1;
jexpiry1=ldeldate-cdate+1;

for jm=1:jexpiry1-1

% determine the payoff to each option in the package
% vall(:,jm,jm+1) is the market forward heat rate for the forward
% maturing on day jm+1 as of day jm.  This assumes that option is
% exercised one day before the delivery date.
```

```
    % Spark spread option payoff (for a call) is the difference between
    % this forward price and the strike price.  vall and strike are
    % measured in MMBTU per MWh.

        payoff(:,jm)=max(vall(:,jm,jm+1)-strike,0);

    end % end jm payoff definition loop

    % pgbeta is the coefficient on the fuel price in the power price
    % equation in the PJ model

        sigmaq2=sigmaq^2;
        sigmaf2=sigmaf^2;
        pgbeta=1;

    % call the implicit solver

        [v,mlarrayx]=fpowdirect_implicit_sparkspread_1(new_qdrift,sigmaq,
          sigmaq2,sigmaf,sigmaf2,rho,pgbeta,q,nq,jexpiry1,dt,dq,payoff);
```

以下是 spark- speed- shell 程序的内置求解程序：

fpowdirect_implict_sparkspread_1.m

```
function [vall,mlarray]=fpowdirect_implicit_sparkspread(drift_mat,sigmaq,
  sigmaq2,sigmaf,sigmaf2,rho,pgbeta,q,nq,nt,dt,dq,pay_mat);

% this function solves the direct problem to value a portfolio of spark
% spread options.
% function returns vall, an array of option values for each maturity, each
% time step, and each load step
% to get value of spark spread option in $/MWh, multiply the elements of
% vall by the appropriate fuel forward price

% drift_mat:=nq x nt matrix of load drift coefficients
% sigmaq:=load volatility
% sigmaf:=fuel volatility
% rho:=load-fuel correlation
% pgbeta:=fuel price exponent in power price equation from PJ model; must
%   equal one in this case to esnure separability of option payoff in load
%   and fuel price
% q:=vector of log load points--this is the load grid
% nq:=number of load points in grid
% nt:=number of time steps until expiration of longest-dated option in the
%   package
% dt:=length of time step--typically 1 day=1/365 years
% dq:=log load step size
% pay_mat:=matrix of payoffs.  Each column gives a payoff for a different
%   option in the package


% create c0 c1 c2 vectors
% these are the coefficients on the first, second, and zero order terms,
% respectively, in the valuation PDE.  This takes advantage of the PJ
% dimenionality reduction that is feasible due to the separability in fuel
% and log load of the spark-spread payoff when pgbeta=1
% due to the time variation and load dependence of these coefficients, each
% is an nq x nt matrix

for jtc=1:nt
```

```
    c1(:,jtc)=pgbeta*rho*sigmaq*sigmaf*ones(nq,1)+drift_mat(:,jtc);
    c2(:,jtc)=.5*sigmaq2*ones(nq,1);
    c0(:,jtc)=.5*sigmaf2*pgbeta*(pgbeta-1)*ones(nq,1);

end

% create al, bl, cl, and ar, br, and cr matrices

nu1=dt/(dq);
nu2=dt/((dq^2));

ar=zeros(nq,nt-1);
br=ar;
cr=ar;

al=ar;
bl=ar;
cl=ar;

centdiff=0;

% user has choice to use central differences, or one-sided differences to
% given the fact that the load mean reversion coefficient is usually very
% large (in absolute value) it is typically desirable to utilize
% non-central differences

for jta=1:nt-1

    if centdiff==0

        c1dum=(c1(:,jta)>0);
        al(:,jta)=-nu2*c2(:,jta)+nu1*(1-c1dum).*c1(:,jta);
        bl(:,jta)=1-dt*c0(:,jta)+2*nu2*c2(:,jta)+nu1*(c1dum.*c1(:,jta)-
          (1-c1dum).*c1(:,jta));
        % use cx instead of cl (c letter l) to avoid confusion
        % with c1 (c one)
        cx(:,jta)=-nu2*c2(:,jta)-nu1*c1dum.*c1(:,jta);

    else


    % central differences

        al(:,jta)=-nu2*c2(:,jta)+.5*nu1*c1(:,jta);
        bl(:,jta)=1-dt*c0(:,jta)+2*c2(:,jta)*nu2;
        cx(:,jta)=-nu2*c2(:,jta)-.5*c1(:,jta)*nu1;

    end

end

% fill in mlarray
% value of the option at time step jt+1 times the appropriate "slice" of
% mlarray equals the value of the option at time step jt (with a greater
% time step indicating a time closer to the present/further from expiration)
% mlarray is nq-2 x nq-2 x nt-1 because the coefficients are time varying

mlarray=zeros(nq-2,nq-2,nt-1);
mrarray=zeros(nq-2,nq,nt-1);
```

```
for jt1=1:nt-1
    for jqr=2:nq-3
        for jqc=1:nq-2;

            if jqc==jqr
                mlarray(jqr,jqc,jt1)=bl(jqr+1,jt1);
            end

            if jqc==jqr+1
                mlarray(jqr,jqc,jt1)=cx(jqr+1,jt1);
            end

            if jqc==jqr-1
                mlarray(jqr,jqc,jt1)=al(jqr+1,jt1);
            end

        end
    end

    % fill in boundary conditions
    % use von Neumann conditions

    mlarray(1,1,jt1)=al(2,jt1)+(4/3)*bl(2,jt1);
    mlarray(1,2,jt1)=cx(2,jt1)-(1/3)*bl(2,jt1);
    mlarray(nq-2,nq-2,jt1)=(4/3)*bl(nq-1,jt1)+cx(nq-1,jt1);
    mlarray(nq-2,nq-3,jt1)=al(nq-1,jt1)-(1/3)*bl(nq-1,jt1);


end


% now begin time stepping
% jt is the time step loop
% jm is the maturity loop

vall=zeros(nq,nt-1,nt);

for jt=nt-1:-1:1
    for jm=jt:nt-1

        if jm==jt


            v(2:nq-1,jm)=mlarray(:,:,jt)\(pay_mat(2:nq-1,jm));
   % apply VN boundary conditions
            v(1,jm)=(4/3)*v(2,jm)-(1/3)*v(3,jm);
            v(nq,jm)=(4/3)*v(nq-1,jm)-(1/3)*v(nq-2,jm);
            vall(:,jm,jt+1)=pay_mat(:,jm);

        else


            v(2:nq-1,jm)=mlarray(:,:,jt)\(v(2:nq-1,jm));
   % apply VN boundary conditions
            v(1,jm)=(4/3)*v(2,jm)-(1/3)*v(3,jm);
            v(nq,jm)=(4/3)*v(nq-1,jm)-(1/3)*v(nq-2,jm);

        end
    end
```

```
    vall(:,:,jt)=v;
end
```

8.9 结论

该模型描述的电力期权价格行为可以通过图表和一些显著结果来更好地理解. 图中所有的期权价值都是建立在 PJ 校准模型的基础上. 这个模型是通过负荷波动率的估计值 σ_q、均值回复参数 k 和用 2000 年 1 月 1 日到 2005 年 5 月 31 日的 PJ 模型数据估计出来的平均对数负荷 $\theta_q(t)$ 来校准的；估计方法的描述见 PJ(2006). 模型是利用 Pirrong-Jermakyan 方法并根据 PJ 电力远期价格（来自纽约商交所清算系统）和 2005 年 6 月 7 日观测到的得克萨斯州东管道区 M-3 的天然气远期价格来进行校准的. 燃料波动率是纽约商交所天然气的平价期货期权合约的内在波动率，这些期权合约的交割月与 2005 年 6 月 7 日观测到的被分析的期权到期日相对应.

估价网格有关于负荷和燃料的 100 个点. 燃料价格最小值为 1.00 美元，最大值为 25.00 美元. 负荷最小值是 1999—2005 年观测到的 PJ 模型最小的负荷，最大值为 2004 年 7 月 15 日（该日期用来确定模型校准过程中的 7 月份远期合约的价值函数，PJ 模型投标数据仅滞后 6 个月）PJ 模型中的负荷投标总额.

图 8.3 描述了 2005 年 7 月 15 日到期的日执行看涨期权在到期日前两天测量的价值，是关于燃料价格和负荷的函数. 期权的执行价是 85 美元，2005 年 6 月 7 日的市价与其相等. 水平维度是燃料价格 f 与负荷 q（图中数据由前至后不断增大）. 如预期一样，期权价值关于燃料价格与负荷递增. 因此，负荷与燃料价格的 δ 值均为正. 也就是说，负荷与燃料价格的 γ 值都为正. 对于高水平的负荷与高燃料价格，负荷的值 γ 尤其大且为正. 这反映了：a）距到期日时间很短时负荷与电力价格关系的凸性；b）期权价值函数的凸性.

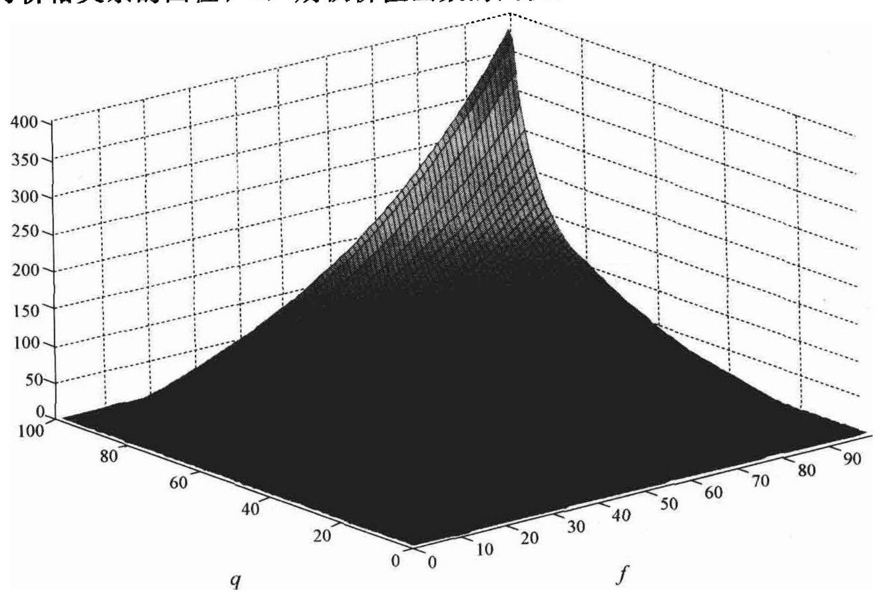


图 8.3 日执行期权价值——2 天到期

图 8.4 描述了相同期权在 2005 年 6 月 7 日或大约到期日前 38 天时的价值. 燃料价格 δ 和 γ 的值从图中看很明显为正；对于燃料中间价格，其凸性尤其明显.

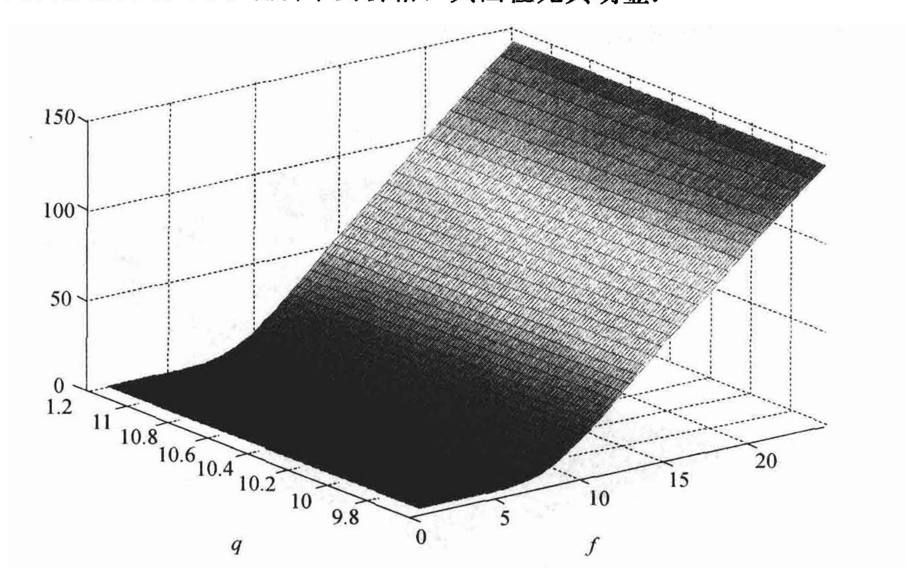


图 8.4　日执行期权价值——38 天到期

然而，期权价值与负荷无关. 事实上，负荷 δ 值与 γ 值都有效为零.（燃料价格给定，把期权价值作为负荷的函数，在负荷区间范围内期权价值的变化比 Matlab 可以描述的最小增量还要小.）在距到期日七八天的时候，负荷的 δ 与 γ 值发生零位调整. 因此，尽管现货电力价格很大程度依赖于负荷，但是到期日大于几天的，日执行期权没有表现出与负荷的相关性.

这一现象反映了负荷很强的均值回复性. 由于均值回复性，受当前负荷限制的未来日期的负荷分布会相当快地收敛于无限制的负荷分布. 因此，对于距到期日超过几天的期权合约，当前负荷的波动对到期时负荷的分布传递的信息很少，因而当前负荷的波动对日执行期权的价值影响很小.

以上分析暗示对于距到期日一周或以上的日执行期权，这种期权是关于燃料的期权. 直至到期日临近，这种期权可以用燃料远期（对冲燃料 δ 值）和燃料期权（对冲燃料 γ 值）来对冲. 在到期日前的最后几天，期权价值逐渐表现出与负荷更强的相关性（尤其是当负荷很高时），套期保值要求利用对负荷敏感的合约（比如，一份远期合约对冲负荷的 δ 值，另一份对负荷敏感的期权合约来对冲负荷的 γ 值).

负荷均值回复对每月交割的电力期权价值的影响尤其明显. 图 8.5 描述了 2005 年 7 月份的月执行看涨期权在到期日前一天的价值. 尽管距到期日如此之短，负荷的 δ 值很小，几乎没有 γ 值. 然而非零的燃料 δ 值和 γ 值很显著. 对负荷的弱相关性反映了月执行期权的价值依赖于期权到期日的平均半个月后交割的远期价格. 除在月执行期权到期日几天后到期的远期合约外，负荷对远期价格几乎没有影响. 因此，到期时负荷的波动对包含在每月集合里的大多数每日远期合约几乎没有影响.

均值回复也对期权的时间衰减有影响. 这对点火价差期权尤其明显. 由于期权可分割成负

荷与燃料的乘积（以及 PJ 框架下这些变量的远期价格的可分性），对点火价差期权价值的限制关于燃料价格是线性的，因此燃料 γ 值为零．所以与月执行和日执行期权形成对照，这意味着点火价差期权的时间衰减不能归因于燃料因素．这种期权的任何时间衰减都应归因于负荷的影响．

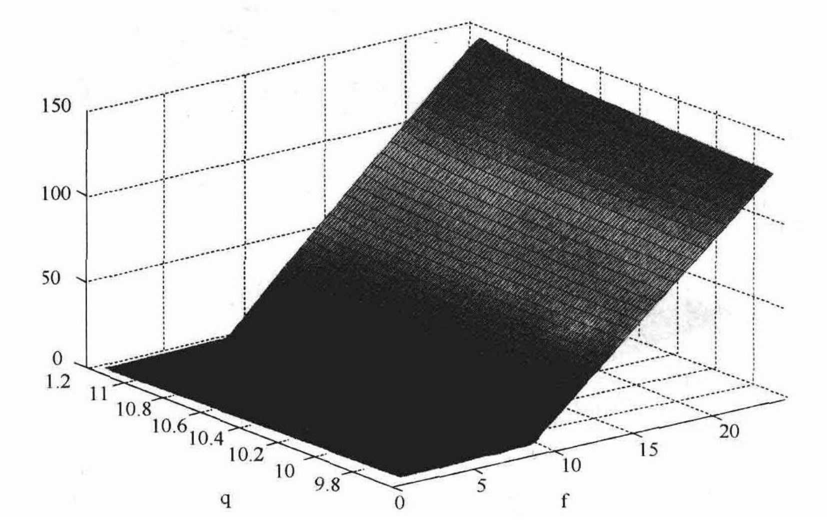


图 8.5　日执行期权价值——1 天到期

考虑到这点，图 8.6 描述了 $H^*=10$ 的点火价差看涨期权的函数 $\Phi(q_t,\ t,\ T)$ 值，Φ 是关于距到期日的时间和负荷（图中有负荷维度）的函数．[19] 图中距到期日的最长时间是 55 天，与 2005 年 8 月中旬的到期日相对应．直到距到期日只有几天的时候期权价值才是稳定的．因此直到临近到期时才有时间衰减．随着期权临近到期，低负荷的期权价值快速下降．相反地，对于高负荷（特别是负荷非常高时），期权价值迅速上升．

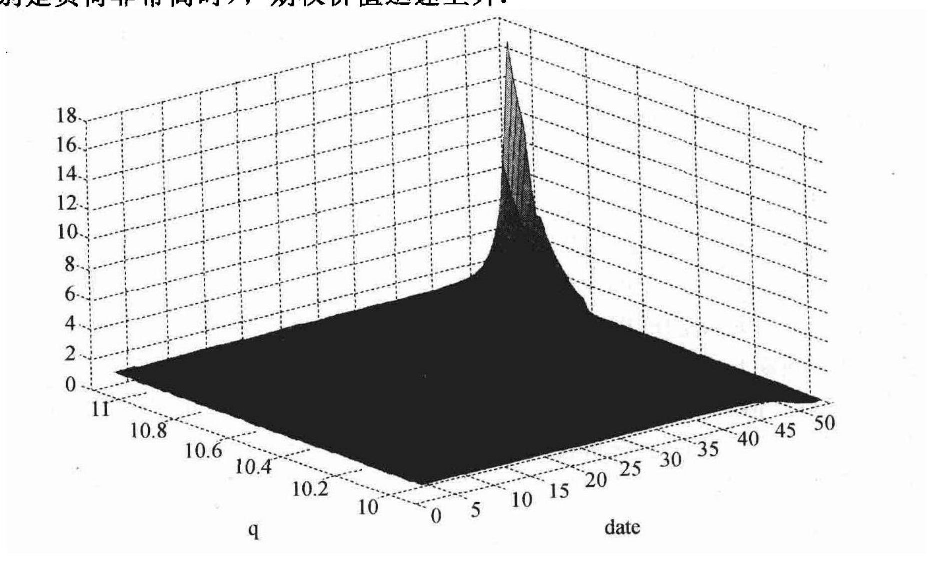


图 8.6　日执行期权价值——55 天到期

这些特征再次反映了负荷的均值回复. 在距到期很久时，由于均值回复，负荷（与点火价差期权价值相关的唯一变量）的条件分布几乎不随时间变化. 这与由几何学上的布朗运动（GBM）决定的期权价值形成对比，在由布朗运动决定价值的期权中，与价值相关的变量的条件分布随时间单调下降. 负荷的稳定性可理解为几乎不存在时间衰减.

相似因素影响日执行和月执行期权的时间衰减. 这些期权表现出时间衰减，但这反映了期权价值依赖于 GBM——燃料价格. 由于到期时燃料价格分布下降，日执行和月执行期权价值分布随时间下降. 使得到期时的燃料价格固定，时间不会影响归因于负荷的期权价值波动. 也就是说，当日执行期权距到期日还有很多天时（不管负荷水平），$\partial\mu/\partial t$ 非常接近于零，即使当月执行期权在即将到期处于等值状态时，$\partial\mu/\partial t$ 还是非常接近于零.

刚发行的电力期权行为应该作为一个警示，要利用标准期权模型来对期权合约进行估价，尤其是那些建立在 GBM 基础上的期权. 非常清楚的是，负荷（或天气）和燃料价格是电力价格短期波动的主要决定因素. 尽管 GBM 假定与价值相关的信息随时间平稳流入，但是由于负荷的均值回复，信息流在临近到期时集中. 本质上，电力期权表现出分散性. 在到期日临近前电力期权是关于燃料的有效期权，但在到期时成为关于负荷的函数. 基于 GBM 的模型不能处理这种跳跃分散性. 跳跃分散式模型能处理这种分散性，但如 8.3 节所述，要描述这种既可控又相对符合现实的模型很困难. 因此，基于基本面的模型具有其重要优点，使得它比其他方法更适合处理电力衍生合约的内在复杂性.

8.10 总结

经过重组后，电力已经成为世界上最大量的商品了. 它也是世界上最复杂的商品，这种复杂性阻碍了其衍生品的发展. 然而电力衍生品市场正在逐步发展，所以对为衍生品估价的合理且可控的模型的需求也相应增长. 电力价格行为的特点使得建立在对电力价格运动外在规定基础上的传统估价方法很有问题. 幸运的是，电力市场基本面的透明性允许一种替代方法的运用. 本章描述的 PJ 模型说明电力价格取决于关键的（且可观测的）供需决定因素，从而得到影响电力衍生品定价的因素. 该模型是可控的，并且可以根据可观测的远期价格用具有理论基础的方法进行修正. 经过修正的模型可以用来为其他的衍生品定价，包括对很多市场参与者来说很重要的数量敏感性合约.

PJ 模型是对电力衍生品定价的一个很好的开端，但是必须认识到它还存在很多不确定的部分，每一部分都可以被改进. 另外，还有待将断电情况和其他电力价格影响因素加入 PJ 模型的基本框架里. 这对期权尤其重要. Pirrong(2006a，2006b) 针对这个问题给出了一些方法，但其他方法可能也是可行的. 因此，基于基本面的电力衍生品估价模型对未来研究是一个很有可能出成果的领域.

尾注

1. 水能发电将储藏因素考虑进电力市场.
2. 尽管电力价格连续，但是电力市场由于电力的不可储藏性也是不完善的. 不可储藏性使得不可能在即期电力市场进行套期保值.

3. 其中一个 α 参数与断电概率向量相乘，可理解为测度变化. 其他的 α 参数不能这样理解.
4. 这一特征内在假定发电系统的物理能力是固定的. 由于新建发电厂与断电情况变化，物理能力允许发生波动.
5. 条件是：1）存在惩罚函数 $h(Q)$，在某个区间是凸的，但在区间外是无限的；2）在不存在任何控制的情况下，Q 服从方程 $\mathrm{d}Q=\mu \mathrm{d}t+\sigma \mathrm{d}W$，惩罚函数可以被理解为高负荷的成本. 如果 $Q>X$，系统会崩溃，从而导致巨额成本. 感谢 Heber Farnsworth 让我知道了 Harrison-Taksar 方法.
6. 这是 Skorokhod 方程的一个例子.
7. Pirrong-Jermakyan(2006) 详细描述了估计 $\theta_q(t)$ 的过程.
8. 如果对负荷有一个更低的限制（负荷最小值），那么存在另一个当地时间过程和 Neumann 类型的边界条件.
9. PJ 有两种方法来建立这种关系：一种利用电力生产商的出价，另一种利用计量经济学手段.
10. 见 Tikhonov& Arsenin (1977).
11. 记住远期价格 F 等于函数 V 与 f^{γ} 的乘积.
12. 价值取决于现货价格的期权存在一些额外的挑战，Pirrong(2006a) 有所描述.
13. 这篇文章的早期草稿中在 $\rho=0$ 的情况下结合求积法和有限次差分法来对期权估价，$\rho\neq 0$ 时用求积法.
14. 这个假设可以被修正.
15. 日执行看跌期权的价值可以类似这样确定.
16. 远期价格可以用根据估价日观测到的远期曲线校准的风险市场价格函数来计算.
17. 对于正的 c1 可以得到相似的表达式.
18. 由于乘积可分性，用 Pirrong-Jermakyan 描述的转换方法可能解出函数 $\Phi(\cdot)$，即使当 $\rho\neq 0$ 时.
19. 当负荷很高时，点火价差期权临近到期时价值极高. 因此为了强调时间衰减的缺乏，避免高负荷的期权价值的影响，只说明负荷不高于均值 15%的点火价差期权价值.

第 9 章　商业房地产资产抵押证券

Tien Foo Sing

9.1　概述

资产证券化是一个多步骤过程，首先要把房地产抵押贷款和其他金融资产汇集在一起，打包成可交易证券，然后通过公开或非公开市场将这些证券出售给投资者. 这种证券被称作*资产抵押证券*（Asset-Backed Security，ABS）. 含有可证券化现金流的房地产及金融资产包括住房及商业抵押贷款、房地产租赁和其他非房地产支持的现金流，如汽车贷款、信用卡贷款、房屋净值信用额度、抵押债务、债券、机器设备租赁和其他贷款及应收账款. 资产证券化已经发展成为一种有效的金融工程技术，直接将资金使用者和提供者通过资本市场联系起来了，因此降低了传统银行贷款中金融中介收取的高额交易成本.

在美国，最早形式的资产证券化是由一个政府赞助机构发起的，政府国民抵押贷款协会（GNMA）在 1970 年首次发行贷款抵押证券. 随后一些其他的准政府或私营机构，如联邦国民抵押贷款协会（FNMA）和联邦住宅抵押贷款公司（FHLMC），也在 20 世纪 80 年代纷纷效仿，将他们的房地产抵押贷款转变成具有很高信用评级的证券. 2005 年的第 4 季度，房地产抵押贷款证券飞跃发展，从零跳跃成一个总市值达 5.9 万亿美元的可观市场.[1] 从市场份额来看，在 2005 年第 4 季度抵押贷款证券构成整个债券市场总债务的 23%（见图 9.1）. 其他应收账款的证券化发展较晚，由电脑设备租赁支持的资产抵押证券开始于 1985 年. 自此之后资产抵押证券市场迅速扩张，从 1985 年的 9 亿美元上升至 2005 年第 4 季度的 1.955 万亿美元. 资产抵押证券市场的主要证券化资产类型如图 9.2 所示.

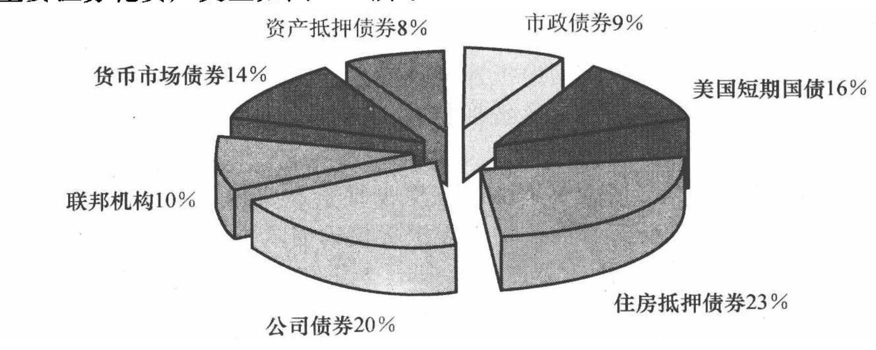


图 9.1　2005 年第 4 季度债券市场

资料来源：债券市场协会.

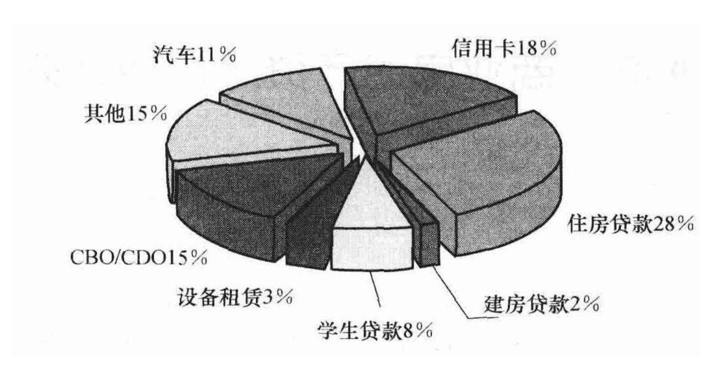


图 9.2 2005 年第 4 季度资产抵押证券市场的主要信用类型
资料来源：债券市场协会.

本章目的是介绍商业房地产支持的证券化，特别是商业房地产出租带来的现金流的证券化，还讨论了房地产生命周期不同阶段的可证券化的现金流，从开发概念和计划的批准、建设，一直到可用楼层的出售或出租.结合现实世界中开发商组织的商业房地产证券化交易来考察新加坡房地产证券化的经验和发展.并描述用互换方法来为这些证券定价，互换方法将证券的违约风险考虑进去了.

9.2 资产证券化的动机

20 世纪 70 年代美国住房抵押贷款证券化是由几个因素推动的.其中一个是美国对提供住房抵押贷款的机构设定存款的利率上限.由于利率上限的设定，当货币市场共同基金和一些替代投资提供比储蓄更高的利率时，储户的资金就从这些机构外流.当储户收回他们的存款时，住房抵押贷款机构会面临流动性压力，而他们不能将长期抵押贷款转变为可以满足储户提款需求的现金.供给和需求现金流在久期上不匹配以及新资本的短缺迫使住房抵押贷款机构缩减贷款.住房抵押贷款成本不断增加对住房所有权产生不利效应.

很多政府和准政府机构，如政府国民抵押贷款协会、联邦国民抵押贷款协会和联邦住宅抵押贷款公司，通过二级市场渠道帮助住房抵押贷款机构填充资本.住房抵押贷款证券化提高了资金从供应方流向需求方的效率.

美国的住房抵押贷款证券市场主要是由于住房抵押贷款机构缺乏新资本供给而开创的.然而在资产证券化市场，由资产所有者的流动性需求促使的需求方因素推动传统贷款市场的金融结构发生变化.证券化提供一个替代选择，减少房地产开发商传统融资对银行和金融中介的过度依赖（见图 9.3).然而从长期看来，如果证券化的效率和成本下降，那么融资安排的转变会导致银行角色的非居间化.当银行和贷款机构作为资本中介的作用减少时，专门的收费服务就会产生，包括资产打包，将现金流构造成可交易证券，信用评级和提高，以及将资产抵押证券出售给投资者.

房地产市场的风险正逐步转移给二级资本市场，银行和金融中介机构给房地产市场提供资金所面临的高度集中的风险就可以减少.通过证券化过程也可以降低资金风险和成本，由于融资渠道的变化，证券化可以为房地产所有者提供更低的融资成本.证券化可以使得资产成为资产负债表外科目.在证券化过程中，他们可以通过将现金流作为特别用途工具出售或分配来实

现账面价值.

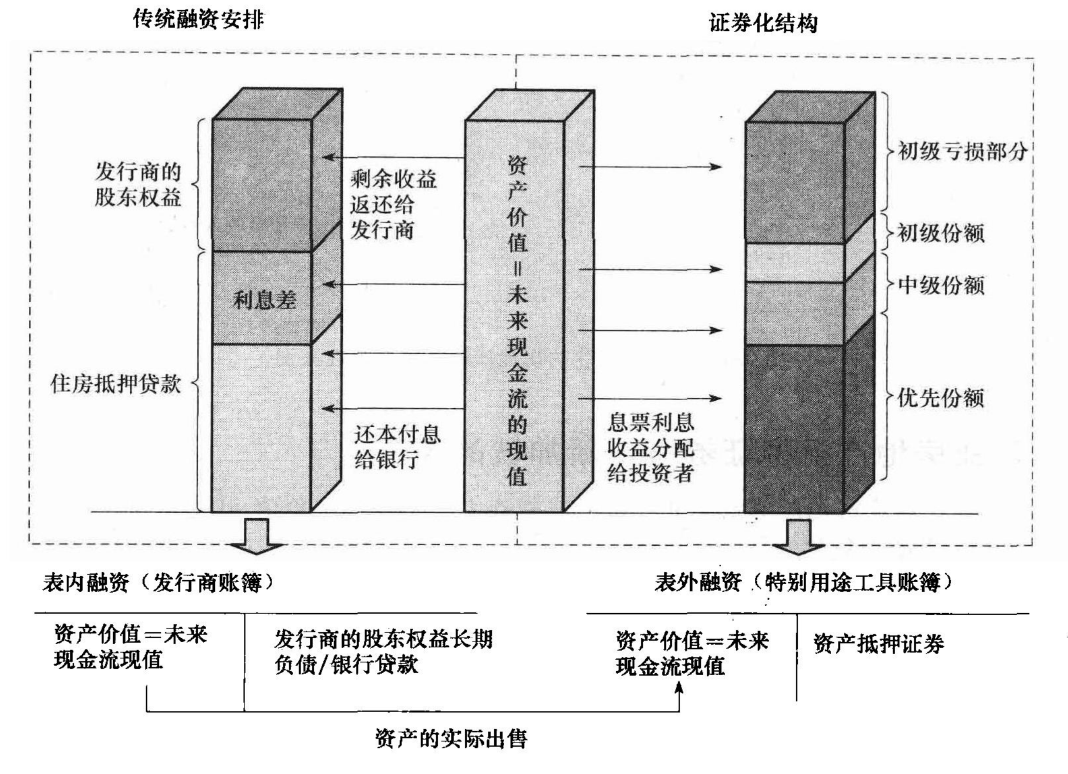


图 9.3 房地产融资的传统市场和证券化市场

从机构投资者的角度看，资产抵押证券是一种新型资产，分散他们投资组合的风险. 这种证券有着与发行者公司风险相分离的投资风险-收益特征.

9.3 房地产现金流证券化的概念

传统房地产融资是资产负债表内业务，作为资产负债表中负债科目下的长期债务，与传统房地产融资不同，证券化涉及将资产打包为特别用途工具（SPV）出售. 如图 9.3 所示，通过出售可以将发行商的长期债务与所有者权益从资产负债表中移除. 房产价值记作资产，负债则被资产抵押证券投资者购买的证券所取代. 这一过程就是*表外融资*.

应用 SPV 来持有这种资产的主要目的是建立一个远离破产的框架，保护投资者规避房地产所有者的经营风险. 通过将证券化资产与所有者的资产负债表分开，投资者有权获得的投资回报形式更简单，直接来自于财产收入. 资产抵押证券的信用评级可以根据资产质量和相应的现金流，与所有者自身的财务和经营风险无关.

证券化技术对不同类型的金融资产普遍适用，只要能够产生未来现金流. 这些现金流可以用各种方式进行金融设计，满足资本市场投资者的风险-收益偏好. 在房地产市场，证券化是根据房地产发展过程完整周期的现金流构造的. 图 9.4 概括了从土地收购、规划阶段至建设，最后到建后阶段产生的现金流. 然而本章仅讨论建后阶段住房抵押商业贷款和租赁的现金流.

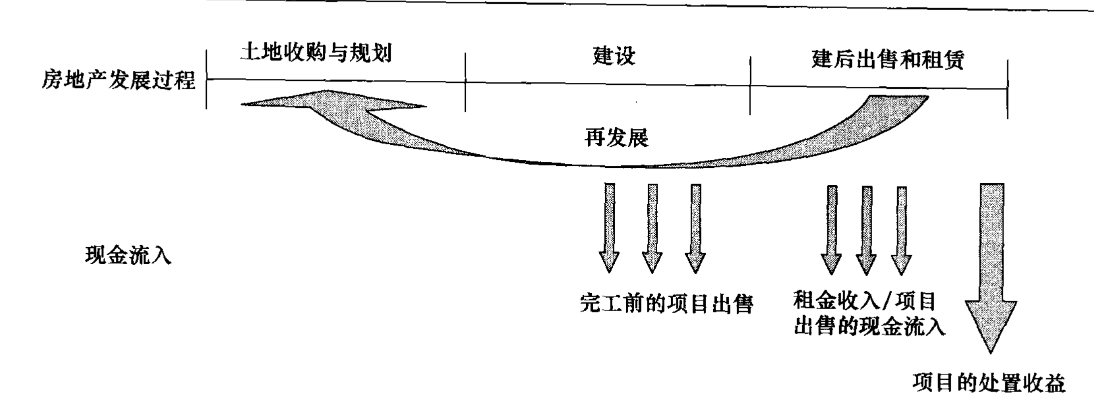


图 9.4 用于证券化的房地产现金流的潜在来源

9.4 商业房地产抵押证券——新加坡的经验

在新加坡，商业房地产贷款的证券化可追溯到1986年.首批价值1169万美元（1850万新加坡元）[2]的住房抵押贷款支持的债券是由Hong Leong Holdings Limited发行的，为其位于市中心商业区的Hong Leong办公楼提供抵押贷款.在1994—1997年间房地产市场的繁荣期，共发行了16种商业抵押贷款抵押债券，总价值14.7亿美元（23.2亿新加坡元）.[3] 新加坡的很多早期证券化交易是被构造为由简单商业资产抵押贷款支持的简单支付债券.证券化为银行提供了一种将商业房地产抵押贷款作为表外业务的方法，降低房地产市场带来的风险.同时，他们通过提供抵押证券打包服务来获得一项新的收入来源.

在1997年重创很多亚洲经济体的亚洲金融危机中，新加坡也没有逃过，很多开发商面临流动性问题，他们的资产账面价值也遭受严重缩水.商业房地产资产的直接出售对开发商来说不是一个可行的选择，因为这要求财产账面价值的严重缩水.另外，由于房地产市场低迷，房地产开发商/所有者很难找到有兴趣批发购买房产的投资者.因此，开发商/所有者利用投资银行结构融资的专业技能，将证券化作为一个二级市场融资的替代方法.这样通过SPV将资产完全出售，从而资产的账面价值没有很大削减.SPV在资本市场通过发行债券来筹集资金，这些债券保证可观的固定息票收益.首批商业房地产支持的证券是由新加坡最大的航运公司——海皇东方航运在1999年发行的，涉及其位于亚历山大路的价值1.1691亿美元（1.85亿新加坡元）的总部大楼的出售.在1999—2002年的3年间，总共有10次商业房地产支持的证券交易，为这些交易提高资金估计发行了25.9亿美元（41亿新加坡元）的债券（见表9.1）.

表 9.1 新加坡商业房地产支持的证券化交易列表

证券化资产	资产类型	发行商/所有者	债券发行量（百万新加坡元）	特别用途工具	承销商	息票率		债券期限	发行日期
						高级债券	初级债券		
海皇航运公司总部	办公室	NOL	185.00	Chenab投资公司	DBS银行	6.75%	7.25%	10年	99-3
Robinson Point	办公室	Birchves投资公司	193.00	Visor有限公司	DBS银行	6.00%	2%	10年	99-7

（续）

证券化资产	资产类型	发行商/所有者	债券发行量（百万新加坡元）	特别用途工具	承销商	息票率		债券期限	发行日期
						高级债券	初级债券		
世纪广场购物中心	购物中心	First Capital	200.00	Pemberton 发展有限公司	DBS 银行	无	无	7 年	99-6
268 Orchard Road	办公室	RE Propeties	184.00	Baronet 有限公司	DBS 银行	5.50%	6.50%	10 年	99-9
Tampines Center	购物中心	DBS 银行	180.00	Tampines 资产有限公司	DBS 银行	5.63%	6.00%	7 年	99-12
Six Battery Road	办公室	Birchves 投资公司	878.00	Clover 有限公司	DBS 银行	6.00%	6.50%	10 年	99-12
Raffles City	购物中心	Raffles	984.50	Tincel 有限公司	DBS 银行	5.00%	7.40%	10 年	01-6
Wisma Atria	购物中心	Al Khaleej 投资	451.00	Upperton (Aspinden) 有限公司	UOB	4.94%	7.0%(A)& 8.85%(B)	5 年	02-5
Compass Point 购物中心	购物中心	Fraser& Neave	335.00	Sengkang Mall 有限公司	DBS 银行	4.88%	8.00%	10 年	02-11
资本广场	办公室	Keppel Land	505.00	Queensley 有限公司	DBS 银行	4.50%	无	7 年	02-11

资料来源：作者编辑.

这些时期发行的大多数商业房地产抵押证券都采用简单的隶属结构，只有债券的两部分——初级和高级部分，不包含其他的第三方担保和信用保护.在这些交易中也没有应用超额担保来提高信用.出售的条款和条件中有回购期权，允许开发商保留享有未来资产升值的收益.回购条款违背了新加坡货币局（事实上的新加坡中央银行）在 2000 年 12 月 4 号发布的 628 通报中制定的法规变化里规定的实际出售要求.因此，一些商业房地产抵押证券的发行商已经将房地产资产重新并入资产负债表.[4]随着法规越来越严格，以及继 2002 年 7 月 CapitalMall 信托公司成功上市后房地产投资信托公司的盛行，房地产投资信托工具已经替代商业房地产支持证券，成为放开发行商资产账面价值的一种更受欢迎的方式.

早期的商业房地产抵押证券不采用集中商业抵押贷款和多层次证券.这两个特性只有当房地产投资信托上市公司将商业房地产组合抵押贷款打包证券化时才能体现出来.CapitalMall 信托公司（CMT）通过特别用途工具，Silver Maple 投资有限公司，利用商业贷款抵押证券结构来获得三部分融资：由 1.72 亿新加坡元的长期贷款、2002 年 2 月 CapitalMall 信托组合的三个零售物业担保的 2 800 万新加坡元的循环信贷额度，以及 2003 年 6 月用于收购 IMM Building 的 1.25 亿新加坡元的长期贷款构成.Ascendas 房地产投资信托公司也将它 40 个物业担保的 6.282 亿新加坡元的抵押贷款转变成商业贷款抵押证券.迄今为止，已经有 6 笔商业贷款抵押证券交易，债券价值总额达 12.9 亿美元（20.4 亿新加坡元）.表 9.2 概述了商业贷款抵押证券交易的细节.

表 9.2 新加坡商业贷款抵押证券交易列表

发行日期	期限	证券种类	发行规模（百万新加坡元）	抵押资产	证券发行商	抵押人
2002-02-26	5 年	两类：利率为3.86%的固定利率债券与利率高于新加坡美元互换利率 0.43%的浮动利率债券①	200.00	CMT 组合的 3 个零售超市：Funan the IT 超市、Junction 8、Tampines 超市	Silver Maple 投资有限公司	CMT
2003-06-26	7 年	高于美元的伦敦银行间同业拆借利率（LIBOR）0.62%的浮动利率债券②	125.00	包含在 CMT 组合中的 IMM Building 收购	Silver Maple 投资有限公司	CMT
2004-02-27	4 年	5 类：A、B、C、D、E（D 和 E 未被评级）③	506.00	3 个零售超市：Reivervale 超市、LotOneShoppers 超市、Bukit Panjang Plaza	CapitaRetail 新加坡有限公司	持有 3 种资产的 Three SinglePurpose 信托公司（SPT）
2004-03-16	5 年	4 类评级债券：A1～A4	580.00	4 幢办公楼，一幢商业和零售的混合楼，两个商业停车场	Silver Loft 投资有限公司	CapitaCommercial 信托公司（CCT）
2004-08-05	5 年	浮动利率债券，欧洲银行间同业拆借利率（EURIBOR）加每年 0.33%的差价④	292.70	17 个物业：6 个商业园区，4 个轻工业工厂，2 家高科技工业企业，5 家物流配送中心	Emerald 资产有限公司	Ascendas 房地产投资信托公司
2005-05-12	7 年	浮动利率债券，欧洲银行间同业拆借利率（EURIBOR）加每年 0.23%的差价⑤	335.40	23 个物业：6 家物流配送中心，6 家高科技工业企业，10 家轻工业企业，1 个商业科技园	Emerald 资产有限公司	Ascendas 房地产投资信托公司

①固定利率票据，如果到 2007 年 2 月 26 号还没被赎回，在 2008 年 8 月 26 的赎回截至日前利率为高于 3 月期新币掉期利率 2.38%.

②浮动利率票据，如果在 2010 年 6 月 26 号前还没赎回，在 2011 年 12 月 26 的赎回截至日前利率为高于美元 LIBOR2.30%.

③CMT 购买 6000 万新加坡元的 CapitaRetail 商业贷款抵押证券的次级债券 E 部分，预期最低收益 8.2%.

④法定到期日是 2009 年 2 月.

⑤法定到期日是 2013 年 11 月 12 号，汇率：1 欧洲美元=2.032 9 新加坡元（2005 年 8 月 23).

资料来源：作者编辑.

商业贷款抵押证券对房地产私募基金、开发商、房地产投资信托公司、二级房地产市场和机构投资者带来了几个好处.对资金使用者（如开发商和房地产投资信托公司）来说，商业贷款抵押证券为他们提供了一个在海外市场获取资金的新渠道，尤其是在对评级高的房地产抵押工具有很大需求的欧洲货币市场.房地产投资信托公司通过商业房地产抵押证券可以保证一个长期稳定的资金来源，并通过资产现金流与债券利息支出配对来对冲利率波动风险.当长期高级债券收益与无风险工具收益相差很小时，资金成本很具吸引力.通过集中抵押贷款，产生规模经济，也同样可以实现资金成本的降低.对二级房地产市场和机构投资者而言，商业贷款抵

押证券给具有固定收益特征的房地产资产类型的选择提供了多样性. 另一方面，债券市场的评级要求与监管机制提高了二级房地产市场的信息效率.

房地产现金流证券化模型从 20 世纪八九十年代早期的单一的贷款抵押债券结构发展成为 20 世纪 90 年代末和 21 世纪初期的更为复杂的结构. 被证券化的是房地产现金流的两部分：股东权益与债务. 然而 2000 年证监会对资产抵押证券的规定更为严格，使得开发商转向房地产投资信托公司对股东权益现金流证券化，再应用资产抵押证券结构. 在抵押贷款方面，商业贷款抵押证券继续吸引着资金使用者，尤其是房地产投资信托公司，将其作为一种对冲利率波动风险的方法. 图 9.5 描述了新加坡商业房地产市场的证券化活动.

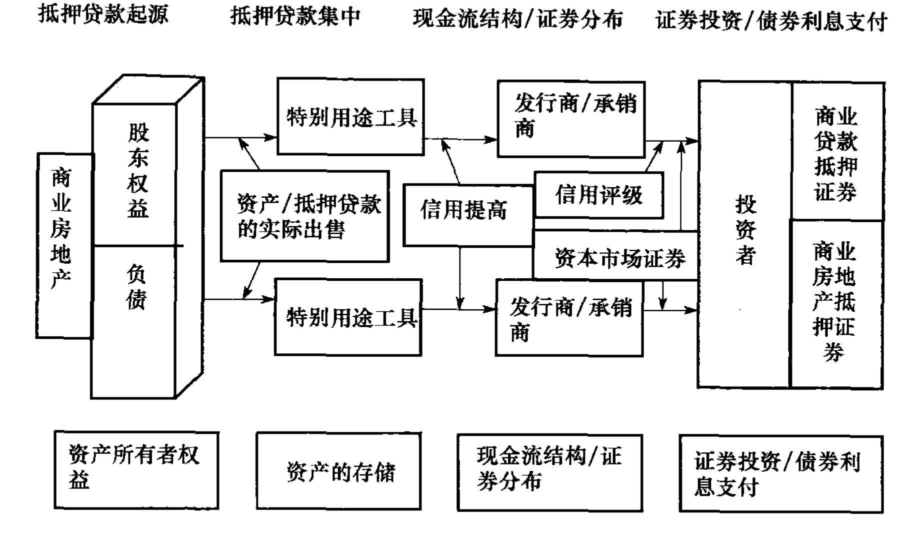


图 9.5 新加坡典型的商业房地产抵押证券

在新加坡，到目前为止证券化主要是由一些资金使用者推动的，包括房地产开发商、商业房地产所有者和房地产投资信托公司. 另一方面，银行也主要关注贷款发行，并且起的作用越来越弱，尤其是在抵押贷款集中、现金流分类和证券构造方面. 展望未来，我们希望商业贷款抵押证券市场能快速发展. 因为借款成本不断上升，新加坡商业银行面临的将在 2007 年全面实行的巴塞尔协议Ⅱ规定的资本充足率要求的压力增加.

9.5 商业房地产抵押证券的典型结构

在资产抵押证券交易中，建立一种特别用途工具（SPV）是为了通过将房地产资产从发行商的资产负债表中剥离来创造一种破产保护结构. 这种结构使得资产抵押证券的投资者不用承担发行商的信用风险. 特别用途工具通过公平分配获得资产的全部所有者权利和义务. 这种安排把它与房地产资产仍保留在发行商资产负债表内的传统抵押贷款和抵押支付证券区分开来. 图 9.6 展示了资产抵押证券交易的结构流程.

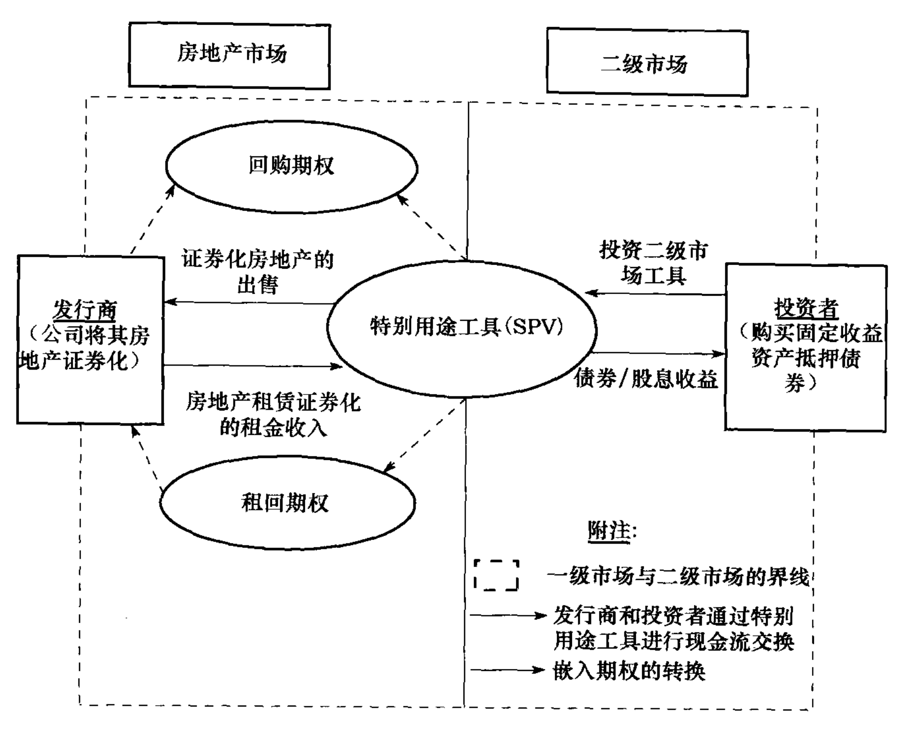


图 9.6 商业房地产资产抵押证券的典型结构

资料来源：Sing、Ong 和 Sirmans (2003)[5].

特别用途工具（SPV）是一个公司法下的法律实体，唯一目标是购买证券化的房地产并在金融市场发行债券和优先股. 对 SPV 的经营活动范围、独立董事的任命以及债务保障政策都有严格的规定. 资产抵押债券是无追索权的债权型证券. 这些规定是为了保证破产隔离，降低 SPV 的破产概率. SPV 不能对债务实行追索，他们仅依赖证券化房地产的现金流来还本付息.

在新加坡，由 SPV 发行的债权型证券，即商业房地产抵押证券，可以分为两类：高级债券和初级债券. 它们对 SPV 累积现金流的要求优先权不同. 高级债券具有更高的优先权与更好的信用质量，通过私募或公开发行的方式出售给偏好有限信用风险的投资者. 然而初级债券由发行商自己保留或出售给要求高风险高收益的企业. 当资产抵押证券被赎回时，高级债券优先于初级债券被清偿.

到目前为止，新加坡发行的商业房地产抵押债券中不存在超额担保的结构. 商业房地产抵押债券的发行总额超过商业房地产的市场价值. 发行商采用租回和购回期权来增强信用. 发行商作为承租人租回证券化的建筑物时，会保证足够偿还债券持有人债务的最低现金流. SPV 也拥有售回期权，使其有权在债券到期时以市场价格将房地产售回给发行商. 这保证了债券在到期时能被完全赎回.

商业房地产抵押债券由在新加坡上市的信誉很好的星展银行承销，这给投资者一个积极信号. 金融机构由于其专业知识和资本市场的经验，能够设计出不同类型的证券及价格，满足投资者的不同投资需求. SPV 正式委托一个资产管理公司（服务商）来管理基础资产的日常经营，收集租金收入，或者为租户提供服务. 除常规的资产管理功能外，SPV 还被要求实施现金

流管理，监督租户服务，监控基础资产状况及报告. 这也保证了租金收入可以作为利息支付给投资者.

Visor 有限公司发行的商业房地产抵押证券案例

在 Visor 有限公司发行的商业房地产抵押证券案例中（SPV 由发行商 Birchvest 私募投资有限公司设立，Birchvest 私募投资有限公司是由 DBS Land（现在的 CapitaLand）完全控股的子公司)，一幢名叫 Robinsin Point、位于罗宾逊路 39 号的办公大楼，在 1999 年 7 月被证券化，价值为 1.219 7 亿美元（合 1.93 亿新加坡元）. 在这项交易中，资产持有公司——Robinson Point Private 有限公司，被其母公司——Birchvest 私募投资有限公司，商业房地产抵押证券交易的发起人，从资产负债表中剥离. 图 9.7 描述了商业房地产抵押证券的结构.

在买卖协议中，发行商承担租回资产的义务，期限为 10 年. 根据这个 10 年期租赁协议，Birchvest 和它的母公司保证支付租金及其他收入，等于或超过 Visor 对债券持有人的利息支付.

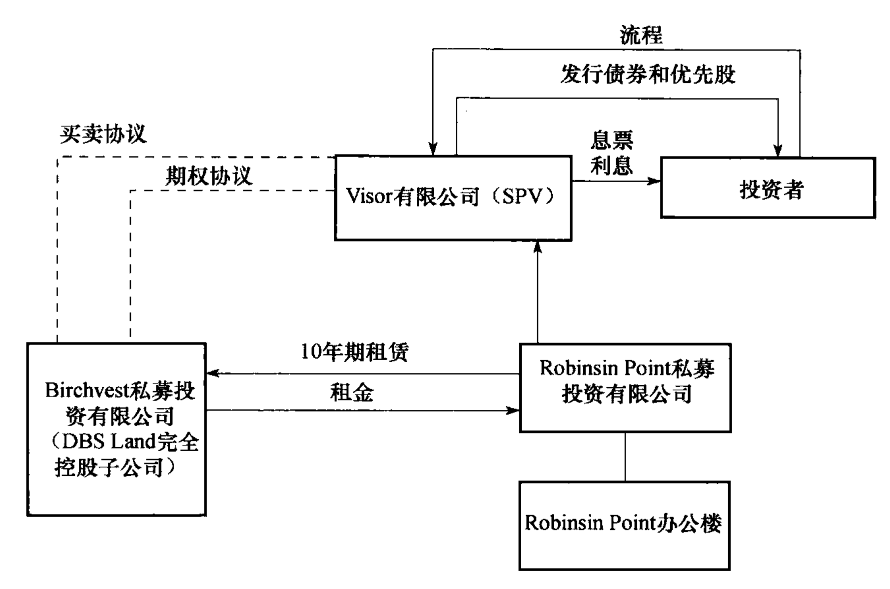


图 9.7　商业房地产抵押证券的典型结构

资料来源：Visor 有限公司商业房地产抵押债券发行招股说明书.

Visor 为购买办公大楼融资，发行了 10 年期固定利率债券和 19 300 份优先股，总计 1.219 7 亿美元（合 1.93 亿新加坡元）. 债券由两部分构成，本金为 7 899 万美元（合 1.25 亿新加坡元）的高级债券与本金为 4 297 万美元（合 6 800 万新加坡元）的初级债券. 高级债券每年支付 2%的固定利率，初级债券每年支付 6%的固定利率，并且两种债券都是延迟半年支付.

在交易协议中，有复杂的购回和售回期权. 发行商被赋予看涨期权，可以在 3 年后在债券到期日 6 个月前行权，购回资产. 在执行购回期权时，如果资产市场价值上升到原来的购买价格 1.219 7 亿美元（合 1.93 亿新加坡元）之上，债券持有人在债券赎回时除收回本金外，还有

权分享超过金额的一部分. 同时，发行商也以售回期权只有在债券到期日前的最后 6 个月才可以行权的形式来向 SPV 保证债券在行权时至少可以平价全部收回.[6]

9.6 商业房地产抵押证券的定价

新加坡的商业房地产抵押证券[7]交易的现金流特征可以与互换合约相比较. 在执行租回期权时，发行商向 SPV 支付类似浮动现金流的市场租金，换得特定租赁期内财产的独家使用权. SPV 将这些浮动现金流转变成债券持有人的固定息票收益. 与互换类似，商业房地产抵押债券持有人从 SPV 那购买现金流（固定利率），而发行商（承租人）向 SPV 支付现金流（浮动利率）. 因此商业房地产证券的信用风险可以在互换框架里衡量，考虑浮动租金、固定息票率、抵押证券名义本金和资产交易价格的不确定性.

9.6.1 互换和互换期权

金融互换 lfinancia（swap）是关于现金支付交换的合约，通常是两方相同名义本金的固定利率支付和浮动利率支付的互换（见图 9.8）. 互换合约对信用风险高度敏感：违约和利率风险.

关于互换的期权，被称作*互换期权*（swaption），授予期权持有者在将来特定时期以特定利率用浮动现金流交换固定现金流的权利，反之亦然. 标准利率互换合约是固定利率和浮动利率支付的交换. 在股票互换中，股票投资者担心股价下降遭受损失，可以签订一份互换协议，同意支付股票收益（股息和资本升值），得到伦敦银行同业拆借利率（LIBOR）或固定利率收益.

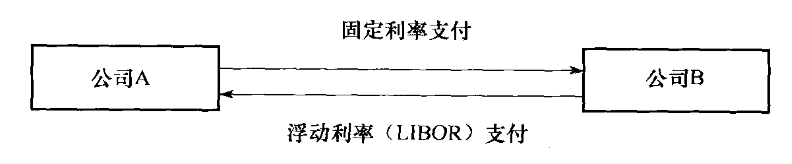


图 9.8 典型的利率互换合同

9.6.2 商业房地产抵押证券的现金流互换结构

本章应用违约风险互换框架来评估商业房地产抵押证券交易的信用风险.[8] 应用互换股价模型来为商业房地产抵押证券的信用风险定价时，可以建立几个假设类简化现金流. 然而这些假设在以下情况可以放宽. 首先，债券的从属结构不复存在，这意味着债券持有人只有一个固定利率的息票收益流入. 其次，发行商在到期之前不执行购回期权，这种债券只能在到期日全部平价赎回. 在临近到期日时，SPV 可以执行看跌期权以等于债券本金的市价将资产卖回给发行商，并且看跌期权协议[9]中规定债券持有人不能分享资产价格升值. 这一假设意味着债券持有人的初始本金会在到期日平价赎回，这避免了为资产价格形成过程建模的复杂性.

由于以上假设，典型的商业房地产抵押证券交易可以被简化为以出租房地产的租金收入形式的浮动现金流来交换对债券持有人的固定利率息票支付的简单互换. 如图 9.9 所示，现金流互换由 SPV 充当中介完成. 由于具有互换的特征，商业房地产抵押证券的具有违约风险的现金流和相关的利率风险可以利用违约风险互换框架很容易地评估.

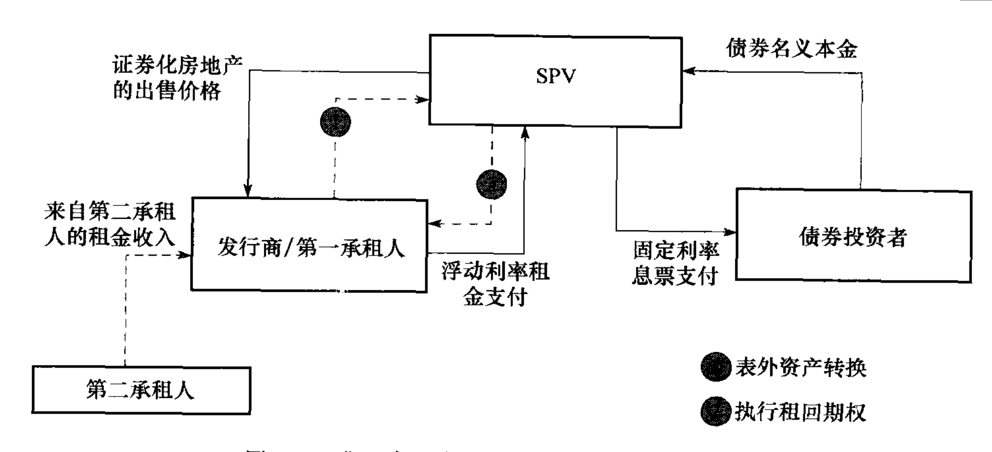


图 9.9　典型商业房地产抵押证券的现金流互换

9.7　利用互换框架为商业房地产抵押证券估价

9.7.1　基本的互换估价框架

理论上，均衡互换利率要使得互换中固定利率和浮动利率现金流的现值相等. 固定互换利率用表示$\bar{r}$，远期利率用表示$\hat{r}$. 两期的固定换浮动无违约风险互换合约的价值由下式决定：

$$b(1)\,\bar{r}+b(2)\,\bar{r}=b(1)\,\hat{r}_1+b(2)\,\hat{r}_2 \tag{9.1}$$

其中，$b(x)$ 表示未来时期的债券贴现价格（名义本金），$x=[1,\ 2]$. 方程（9.1）左右两边同除以［b(1)+b(2)］，均衡固定互换利率$\bar{r}$可以估计为远期利率的凸性组合：

$$\bar{r}=\frac{b(1)}{b(1)+b(2)}\,\hat{r}_1+\frac{b(2)}{b(1)+b(2)}\,\hat{r}_2 \tag{9.2}$$

如果不考虑违约风险，方程（9.1）的互换估价可以进一步一般化为多期（N 期）互换，如下所示：

$$\sum_{i=1}^{N}b(i)\,\bar{r}=\sum_{i=1}^{N}b(i)\,\hat{r}_i \tag{9.3}$$

方程（9.2）中互换合约的均衡固定利率可以表示成远期利率的加权平均，如下所示：

$$\bar{r}=\frac{\sum_{i=1}^{N}b(i)\,\hat{r}_i}{\sum_{i=1}^{N}b(i)} \tag{9.4}$$

9.7.2　运用互换模型为商业房地产抵押证券的信用风险定价

商业房地产抵押证券的发行商执行租回期权时，他就拥有在债券到期之前一段时期的房地产的独家使用权. SPV 则根据租赁协议向发行商收取使用房地产的租金. 租金现金流转变成固定利率息票支付给债券持有人. 技术上来看，SPV 通过从发行商收到浮动现金流支付给债券持有人固定利率现金流来完成互换.

均衡时，支付给债券持有人的固定利率息票应该等于出租证券化资产的现金流的现值，使得在不存在违约风险情况下互换价值为零. 无违约风险的均衡互换价值可以由方程（9.5）表示，如下所示：

$$\sum_{i}^{N} b(i) FRP + b(N) V_0 = \sum_{i}^{N} b(i) NRI_i + b(N) V_N \tag{9.5}$$

其中，FRP 是固定利率支付，NRI_i 是 i 期的租金净收入，V_0 是 SPV 发行的债券本金，V_N 是证券化资产在时间 N 的价格，与债券到期日正好相等.

方程（9.5）左右两边同时除以 V_0，该方程可以重写为：

$$\sum_{i}^{N} b(i)\, \bar{r} + b(N) = \sum_{i}^{N} b(i) R_i + b(N) \delta \tag{9.6}$$

其中，$\bar{r}$表示债券的固定息票率，R_i 表示浮动租金收入，δ 表示到期时房地产价格与债券本金的比率.

9.7.3 商业房地产抵押证券互换中违约风险的建模

Duffi 和 &Singleton(1997)[10]模型中违约是一个外生过程，由危险率函数和损失弥补定义，这是违约情况下债券市场价值的一部分. Duffie 和 Singleton(1997) 的违约债券定价模型可表示如下：

$$V(t) = E_Q\left[\exp - \left(\int_r^T R(s) \mathrm{d}s\right) \Big| \zeta_t\right] \tag{9.7}$$

其中，E_Q 表示在到 t 期存在的信息，ζ_i 的风险中性期望条件，$R(t)$ 表示违约调整的短期利率，是短期无风险利率与违约风险溢价之和.

在具有违约风险的互换模型中，贴现率变量将不确定性考虑进去了. 调整后的租金回报率（R）是第一个状态变量，被认为遵循一个带有均值回复趋势的特定随机过程，如下所示：

$$\mathrm{d}R = \alpha(\mu - R)\mathrm{d}t + \sigma_R R \mathrm{d}z_1 \tag{9.8}$$

其中，α 为租金调整速度，μ 为长期平均租金，$\sigma_R{}^2$ 是租金变化的瞬时方差系数，$\mathrm{d}z_1$ 是标准 Wiener 过程的增量. [$\alpha(\mu-R)$] 是漂移项，代表促使租金回报向长期价值收敛的均值回复力. 式子右边第二项用"R"而不用"$\sqrt{R}$"，表示标准扩散过程，足够灵活可以掌控负的租金回报率.

第二个状态变量是无风险贴现率（r），遵循标准的 Cox、Ingersoll 和 Ross（CIR）(1985)[11]随机过程，可表示为：

$$\mathrm{d}r = k(\theta - r)\mathrm{d}t + \sigma_r \sqrt{r} \mathrm{d}z_3 \tag{9.9}$$

其中，θ，k，δ_r 为正的常数，$\mathrm{d}z_3$ 为标准 Wiener 过程.

基于之前的假设，无违约风险的利率互换可表示为一个债券组合，由固定（浮动）利率债券的多头头寸和浮动（固定）利率债券的空头头寸构成. 它可以作为一系列到期日为对应的互换合约中规定的交割日期的远期利率协议（FRA）[12]被评估. 由于远期利率协议的利率长期结构，互换的固定利率可以通过互换的初始价值为零来确定. 然后均衡的无违约风险现金流互换可以通过固定利率和浮动利率现金流相等来确定，如下所示：

$$\sum_{i=1}^{N} b_x(0,i)FRP + b_x(0,N)V_0 = \sum_{i=1}^{N} b_l(0,i)NRI_i + b_l(0,N)V_N \tag{9.10}$$

其中，$b_x(0,\ i)$ 和 $b_l(0,\ i)$ 分别代表按固定利率和浮动利率贴现的债券价格.

用传统鞅方法（由下标 Q 表示）来解方程（9.10），可以得到如下互换定价方程：

$$E_Q\left\{\sum_{i=1}^{N} FRP\exp\left[-\int_0^i r_i^c(s)\mathrm{d}s\right]\middle|\zeta_i\right\} = E_Q\left\{\sum_{i=1}^{N} NRI_i\exp\left[-\int_0^i R_i(s)\mathrm{d}s\right]\middle|\zeta_i\right\} \tag{9.11}$$

其中，r_i^c 表示交换债券的利率，由无风险利率（r_i）和固定的信用溢价（h）构成，即 $r_i^c=r_i+h$，R_i 表示租金回报率，ζ_i 表示到 i 期为止所有被披露的信息.信用风险溢价通过固定息票收益和无风险利率之差来衡量，用来补偿固定利率收入者（债券持有人）承担的违约风险.它考虑了违约概率及违约的损失补偿.

用 $V_{NRI}(t)$ 和 $V_{FRP}(t)$ 分别表示商业房地产抵押证券交易的浮动利率和固定利率支付的现值.资产抵押证券的基本互换定价公式的违约风险部分可表示为：

$$V(t) = V_{NRI}(t) - V_{FRP}(t). \tag{9.12a}$$

$$V(t) = E_Q\left\{\sum_{i=1}^{N} NRI_i\exp\left[-\int_0^i R_i(s)\mathrm{d}s\right]\middle|\zeta_i\right\} - E_Q\left\{\sum_{i=1}^{N} FRP\exp\left[-\int_0^i r_i^c(s)\mathrm{d}s\right]\middle|\zeta_i\right\} \tag{9.12b}$$

其中，$V(t)<0$ 表明债券持有人易受发行商的违约风险影响.然而，当 $V(t)\geqslant 0$ 时交易对债券持有人有利.

9.8 典型商业房地产抵押证券违约风险的数值分析

方程（9.11）与方程（9.12b）给出的违约风险互换模型没有包含标准解析解.可以应用蒙特卡罗模拟技术来联合确定补偿违约风险浮动利率部分的互换价值和均衡互换利率.为了更好地说明，违约风险互换模型被应用于一个实际案例——Visor 商业房地产抵押证券，它在“Visor有限公司的商业房地产抵押证券案例”中有所描述.发行商 Birchvest Private 有限公司将 Robinson Point 办公大楼证券化，发行证券价值为 1.219 7 亿美元（合 1.93 亿新加坡元）.

9.8.1 蒙特卡罗模拟过程

随机贴现率过程是用蒙特卡罗模拟技术来形成的.首先产生的是方程（9.8）与互换模型简化形式定义的随机状态变量的离散值.运用欧拉离散方法，租金回报和无风险利率的连续时间随机过程可以如下估计：

$$\Delta R_k = \alpha(\mu - R_{k-1})\Delta t + \sigma_R R_{k-1}\sqrt{\Delta t}z_1 \tag{9.13}$$

$$\Delta r_k = k(\theta - r_{k-1})\Delta t + \sigma_r\sqrt{\Delta r_{k-1}\Delta t}z_2 \tag{9.14}$$

服从二元标准正态分布的随机样本 z_1、z_2 被定义为：

$$z_1 = \varepsilon_1 \tag{9.15}$$

$$z_2 = \rho\varepsilon_1 + \varepsilon_2\sqrt{1-\rho^2} \tag{9.16}$$

其中，ρ 是二元分布中两个变量的相关性，ε_1 与 ε_2 是从一元标准正态分布中独立得到的.模拟

中离散时间间隔为 6 个月，即 $\Delta t=0.5$.

得到从零时刻至债券到期日 $t=[0, k]$ 的租金回报 R_t，与无风险利率 r_t 的随机值，遵循方程（9.13）与方程（9.14）的离散随机过程. 利用样本可以计算得到浮动利率和固定利率现金流的贴现价值.

当运用欧拉方法得到方程（9.14）给出的无风险利率随机变量的离散值时，无风险利率路径是通过 k 个正态分布变量 $\{\hat{r}_0, \cdots, \hat{r}_k\}$ 得到的. 用这 k 个变量产生的无风险利率贴现就可以得到债券价格，如下所示：

$$\exp\left(-\sum_{j=1}^{N}(\hat{r}_j+h)\Delta t\right) \tag{9.17}$$

k 维无风险利率路径的模拟被重复 M 次. 用 $f^n(\Delta t)$ 表示路径 n 的债券现金流. 经蒙特卡罗模拟得到的 k 维无风险利率贴现的风险现金流的平均现值可以被表示为：

$$\frac{\sum_{n=1}^{M} f^n(\Delta t)\exp\left[-\sum_{j=1}^{N}(\hat{r}_j+h)\Delta t\right]}{M} \tag{9.18}$$

蒙特卡罗模拟过程然后被再次用于风险租金（浮动）现金流.

9.8.2 参数输入

根据表 9.3 概括的一系列参数输入，商业房地产抵押证券的违约风险互换价值可以通过方程（9.12b）计算得到. 一些参数，如租金波动率（σ_R）、无违约风险利率波动率（σ_r）、平均租金率（μ）、无风险利率（θ）以及租金率和无风险利率的相关系数（ρ），都是根据新加坡相关公共来源得到的历史数据事后确定的，包括市区重建局（URA）的季度租金指数和从 1990 年到 2002 年的新加坡货币局 5 年期债券的季度收益率.

表 9.3 蒙特卡罗数值分析的输入参数假设

输入参数	基　值
A）历史估计参数：	
证券化房地产的售价	$V_0=193\,000\,000$ 美元
初始租金率	$R_0=7\%$
长期平均租金率	$\mu=-1.31\%$
年度租金波动率	$\sigma_R=14.02\%$
固定利率息票收益	$r^c=5\%$
初始（随机）无违约风险的利率	$r_o=3.7\%$
平均 5 年无违约风险的利率	$\theta=3.78\%$
无违约风险的利率波动率	$\sigma_r=0.83\%$
随机利率和租金率的相关性	$\rho=-15.9\%$
B）校准参数：	
信用溢价	$h=5.3\%$
租金回报均值回归速度	$\alpha=8\%$
无风险利率调整速度	$\kappa=40\%$

两个均值回复调整参数（α 和 k）和信用溢价参数（h）是通过将互换方程（9.12）的两个具有违约风险的现金流的现值平衡为零校准得到的.

9.8.3 结果分析

根据前面的输入参数，用 Matlab 进行蒙特卡罗模拟，程序代码在下一节给出. 表 9.4 对模拟结果进行了概括.[13]如表 9.4 所示，具有违约风险的互换价值被定义为浮动利率现金流和固定利率现金流经违约调整后的短期利率贴现得到的现值之差.[14]当互换的违约风险部分价值为正时，浮动利率现金流的累积现值高于固定利率现金流的累积现值. 租金收益足够支付债券利息. 这意味着浮动利率现金流的支付者（发行商）的违约概率很低. 互换的违约风险部分价值越大，对固定利率收入者（债券持有人）的保护程度越高，即更低的违约风险. 当具有违约风险的互换价值为负时，租金收益很可能不够用来支付债券利息. 债券持有人承担高违约风险.

对于一个确定的情况，租金率波动率和无风险利率都为零，互换合约的违约风险溢价估计值为 468 万美元（合 740 万新加坡元），相当于证券化房地产初始价值的 3.83%.

表 9.4 互换贴现价值的蒙特卡罗模拟结果

租金率波动率	无违约风险的利率波动率	互换价值		
		百万美元#	百万新加坡元	(%) @
A) 确定性情况				
0%	0%	4.68	7.4	3.83%
B) 随机情况				
10%	5%	4.33	6.85	3.55%
	10%	4.27	6.75	3.50%
	20%	4.25	6.73	3.49%
	30%	4.63	7.32	3.79%
	40%	5.11	8.09	4.19%
	50%	5.55	8.79	4.55%
20%	5%	3.41	5.4	2.80%
	10%	3.22	5.09	2.64%
	20%	3.32	5.26	2.73%
	30%	3.72	5.89	3.05%
	40%	4.26	6.74	3.49%
	50%	4.60	7.28	3.77%
30%	5%	1.70	2.69	1.39%
	10%	1.64	2.6	1.35%
	20%	1.80	2.85	1.48%
	30%	2.19	3.46	1.79%
	40%	2.57	4.07	2.11%
	50%	2.94	4.66	2.42%

（续）

租金率波动率	无违约风险的利率波动率	互换价值		
		百万美元	百万新加坡元	(%) @
40%	5%	−0.67	−1.06	−0.55%
	10%	−0.83	−1.32	−0.68%
	20%	−0.54	−0.85	−0.44%
	30%	−0.11	−0.17	−0.09%
	40%	0.13	0.2	0.10%
	50%	0.62	0.98	0.51%
50%	5%	−3.44	−5.45	−2.82%
	10%	−3.68	−5.83	−3.02%
	20%	−3.43	−5.43	−2.81%
	30%	−3.32	−5.25	−2.72%
	40%	−2.88	−4.55	−2.36%
	50%	−2.32	−3.67	−1.90%

注：@互换价值估计值占债券发行名义价值（1.93 亿新加坡元）的百分比.
根据 2006 年 4 月 29 日的汇率，1 美元＝1.5824 新加坡元.

根据历史数据，新加坡的年度租金波动率大约是 14%，所以具有违约风险的互换价值总是为正，这意味着固定利率收入者承担的违约风险被适当地降低了. 只有当租金波动率超过30%时，具有违约风险的互换价值才会是负值.

9.9 数值分析的 Matlab 代码

以下 Matlab 代码是为商业资产抵押证券定价的蒙特卡罗模拟过程.

exrelationdepend25.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Monte Carlo Simulation for CREBS valuation using the proposed swap
%   pricing model
%
%   Variable Description:
%
%   V0= the current property asset value
%   u1= the long-term average property return rate
%   a1= the speed of property return rate adjustment
%   u2= the long-term average fixed interest rate
%   a2 = the speed of risk-free interest rate adjustment
%   rho= correlation
%   nSimulations = Number of simulations used for Monte Carlo simulation
%   h= credit spread
%   More simulations increase accurancy, typically minimum 10000
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function S=exrelationdepend25(V0,u1,u2,a1,a2,r,nSimulations,rho,h)

       floatvol=0:0.05:0.5;
       fixvol=0:0.05:0.5;

       num_p=length(floatvol);
       num_r=length(fixvol);
```

```
        exchangeprice=zeros(num_p,num_r);

         for k=1:num_p
           for l=1:num_r
              cash1=0;
              cash2=0;
              sum5=0;
              sum6=0;
              for j = 1: nSimulations
               sum3=0;
               sum4=0;
               Rt=0.07;
               rt=0.037;

               for i=1:20;

                  Epsilon3 = normrnd(0,1);
                  Rt = Rt + a1*(u1-Rt)*0.5 + floatvol(k)*sqrt(0.5)*
                 Rt*Epsilon3;
               sum3=sum3+Rt*0.5;
               discount1=exp(-sum3);
               sum5=sum5+discount1*Rt*V0*0.5;

               Epsilon4 = rho * Epsilon3 + normrnd(0,1) * sqrt(1 -
                 rho ^ 2);
               if rt<0
                  rt=0;
               end
               rt = rt + a2*(u2-rt)*0.5 + fixvol(l)*sqrt(0.5*rt)*
                 Epsilon4;
               sum4=sum4+0.5*(rt+h);
               discount2=exp(-sum4);
               sum6=sum6+discount2*r*V0*0.5;

            end
       end
               cash1=sum5/nSimulations;
               cash2=sum6/nSimulations;

           exchangeprice(k,l)=cash1-cash2;
    end
 end
 S=exchangeprice;

 surf(fixvol,floatvol,exchangeprice);
 xlabel('Riskless Interest Rate Volatility');
 ylabel('Commercial Property Rent Volatility');
 zlabel('Swap Present Value');
 title('Sensitivity Measures');
 axis([0 0.5 0 0.5 -inf inf]);
 set(gca,'box','on');
```

```
>> V0=193000000;
```

假如用下列参数：

```
u1=-0.013;
u2=0.038;
a1=0.01;
a2=0.3;
r=0.05;
nSimulations=50000;
```

```
rho=-0.159;
h=0.053;
```

运行模拟过程，交易价格计算如下：

```
>> S=exrelationdepend25(V0,u1,u2,a1,a2,r,nSimulations,rho,h)
S = 1.0e+006 *
```

图 9.10 显示模型 A 的单方违约模型中的资产（房地产）互换的现值.

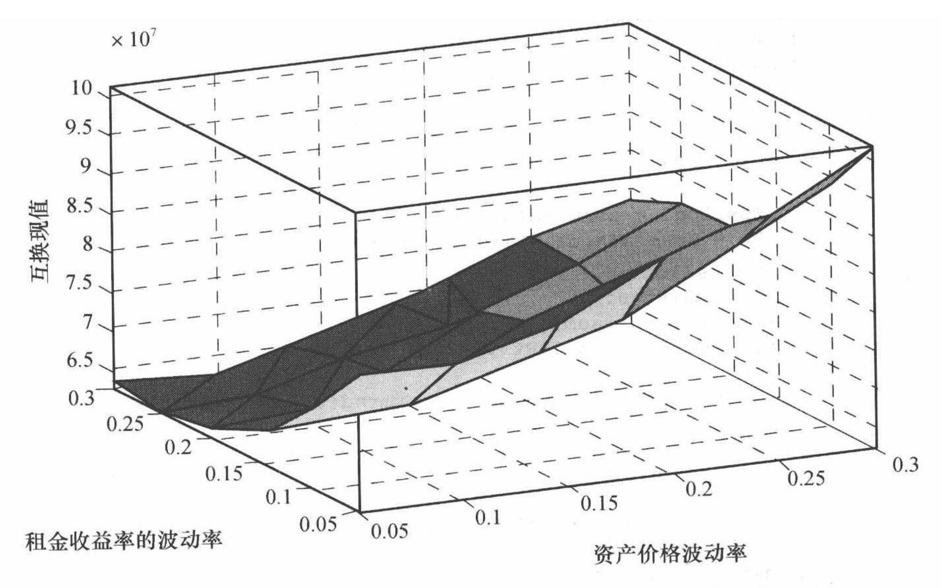


图 9.10 模型 A 的单方违约模型中的资产（房地产）互换的现值

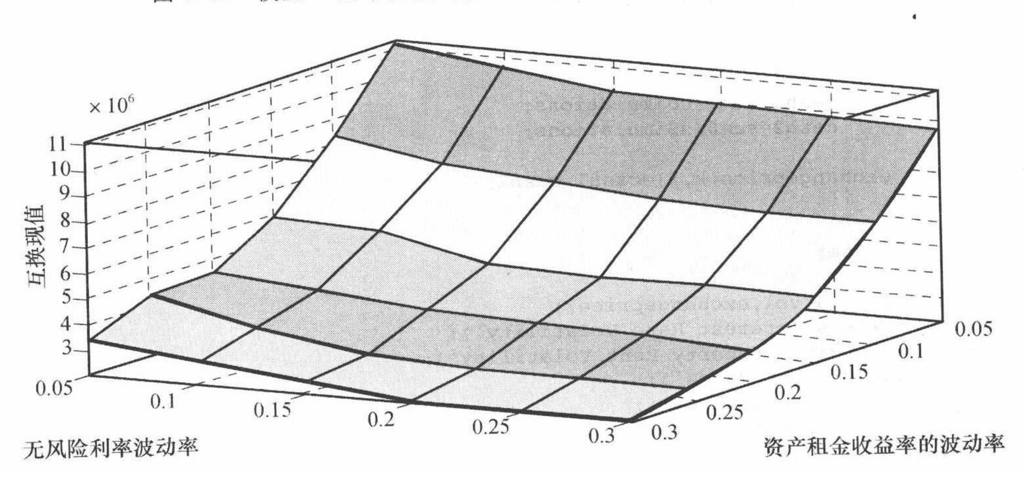


图 9.11 模型 B 中存在双方违约风险的互换现值

9.10 总结

在美国，商业抵押贷款证券化市场迅速发展，几乎每 5 笔商业抵押贷款中就有一笔被证券

化.提供抵押贷款的机构和商业银行都很积极地把他们的商业抵押贷款打包，然后在资本市场出售，作为一种注入新的营运资本的方式.然而在新加坡，商业银行对于将商业抵押贷款证券化出售的想法一直以来都不是很积极.目前健康的流动资金状况、商业抵押贷款的低违约风险，以及银行不愿意破坏与抵押贷款企业的良好关系都使银行没有发行商业贷款抵押证券的动机.巴塞尔协议Ⅱ对银监会风险资本充足管理的规定一旦在2007年末在新加坡全面实施，可能会促使商业银行将证券化作为一种对冲商业抵押贷款信用风险的方式.

对于房地产资金使用者来说，商业房地产抵押证券和商业贷款抵押证券仍然是一种很具吸引力的筹资渠道，其潜力还没有被完全开发出来.当证券化技术在股票市场已经司空见惯时，住房抵押贷款首要供应商，如商业银行和金融机构的中介化，在将来也是可能的.

本章用互换定价模型来为商业房地产抵押证券的违约风险部分定价.结果显示租金率波动与无风险利率是决定商业房地产抵押证券交易中违约风险价值的两个重要变量.设定商业房地产抵押证券交易的均衡固定利率和浮动利率时，将证券化房地产的租金收益和无风险利率的波动考虑进去以减小违约风险是很重要的.

尾注

1. 来源：债券市场协会.
2. 根据2006年4月29号的汇率，1美元=1.5824新加坡元.
3. Ong S E, Ooi J, and Sing T F(2000), "Asset Securitization in Singapore: A Tale of Three Vehicles." *Real Estate Finance*, 17(2), pp. 47-56.
4. Fan G Z, Sing T F, Ong S E, and Sirmans C F(2004), "Governance and Optimal Financing for Asset-Backed Securitization", *Journal of Property Investment & Finance*, Vol. 27, No. 5, pp. 414-434.
5. Sing T F, Ong S E, and Sirmans CF(2003), "Asset-Backed Securitization in Singapore: Value of Embedded Buy-Back Options", *Journal of Real Estate Finance & Economics*, Vol. 27, No. 2, pp. 173-180.
6. 要了解关于期权特征和定价的更多细节，参考 Sing T F, Ong S E, &Sirmans C F(2003), "Asset-Backed Securitization in Singapore: Value of Embedded Buy-Back Options", *Journal of Real Estate Finance & Economics*, Vol. 27, No. 2, pp. 173-180.
7. 具有违约风险的互换定价方法基于下面论文：Sing T F, Ong S E, Fan G Z, and Sirmans C F(2004), "Analysis of Credit Risks in Asset-Backed Securitization Transactions in Singapore", 作为2002年马斯特里赫特-剑桥研讨会的特殊问题讨论，*Journal of Real Estate Finance & Economics*, Vol. 28, No. 2/3, pp. 235-254.
8. Duffie&Huang(1996), Duffie&Singleton(1997), Hubner(2001) 在以下论文中提出了互换估价模型：Duffie, D. and M. Huang(1996), "Swap Rates and Credit Quality", *Journal of Finance*, 51(3), July, pp. 921-949. Duffie D and K Singleton(1999), "Modeling Term Structures of Defaultable Bonds", *Review of Financial Studies*, 12(4), pp. 687-720. Hubner G(2001), "The Analytic Pricing of Asymmetric Defaultable Swaps", *Journal of Banking & Finance*, 25, pp. 295-316.

9. 资产抵押证券发行商必须向SPV支付初始市场价格加上售回期权执行时一定百分比的资本利得. 这篇论文省略了分配给债券持有人的资本利得部分. 不过该省略不会显著影响分析结果.
10. Duffie D, K Singleton(1997), "An Economic Model of the Term Structure of Rate Swap Yields", *Journal of Finance*, 52(3): 1287 - 1321.
11. Cox J C, Ingersoll J, Ross S(1985), "A Theory of the Term Structure of Interest Rates", *Econometrica*, 53, P385 - 407.
12. 远期利率协议（FRA）是单期双边协议，一方同意在将来按照一段固定利率时期计算，收取或支付事先确定的固定利率（协议利率）与LIBOR之差.
13. 作者希望向Dr. Fan, Gangzhi表示感谢，感谢他在用Matlab进行数值分析方面的帮助.
14. 经违约调整后的短期利率是无风险利率与违约风险溢价之和，这与风险中性假设一致. 用违约调整后的短期利率对现金流贴现要考虑违约的概率和时间，以及违约的损失补偿(Duffie&Singleton, 1999).

附录 A　Matlab 中的利率树建模

本附录提供用于固定收益债券定价领域中常用模型的 Matlab 实现，这些模型包括：Hull-White模型、Black-Derman-Toy（BDT）模型、Black-Karasinski 模型、Heath-Jarrow-Morton（HJM）模型. 本书提供了构建树、定价和计算 Greek 敏感性的 Matlab 代码. 事实上，为定价利率衍生品，我们通过测量利率树获得市场数据，即收益和波动性的期限结构，从而为利率衍生品定价. 由于本章注重构建树的 Matlab 实现，因此没有介绍构建树程序的细节、潜在理论和模型的来历，读者可以参考其他资料.

A. 1 节介绍了在 Matlab 中如何构建 BDT 树，并给出了固定收益债券定价的例子. A. 2 节介绍如何在 Matlab 中构建 Hull-White 树. A. 3 节展示了在 Matlab 中如何构建 Black-Karasinski 树以及如何用 Matlab 的 treeviewer 展示这些树. A. 4 节介绍 Matlab 中 HJM 定价. A. 5 节提供了 Matlab Excel 连接的例子，来说明 Matlab 如何与 Excel 交互用 HJM 模型为息票债券定价. A. 6 节给出了 2 因素 HJM 模型在 Matlab 中的实现.

A. 1　Matlab 中的 BDT 建模

BDT 模型背后的思想是利率以 1/2 的概率升高或降低. BDT 树的构建是 1991 年由 Jamshidian 采用前向归纳技巧开发出来的.[1] Jamshidian 表明在 BDT 模型里，t 时刻的短期汇率水平服从一个对数的正态随机过程，表示如下：

$$r(t) = U(t)\mathrm{e}^{(\sigma(t)z(t))} \tag{A. 1}$$

这里，$z(t)$ 是一个标准布朗运动过程，$\sigma(t)$ 是 t 时刻的短期浮动汇率，$U(t)$ 是 $r(t)$ 所服从分布的中位数. 每一个时间步都要确定 $U(t)$ 和 $\sigma(t)$ 使模型与观测到的收益和波动性曲线相匹配. 如果假设 $\sigma(t)$ 是常数，这样构建模型只需要匹配收益曲线，则只需确定 $U(t)$ 即可. 在这种情况下，t 时刻的短期汇率水平简化为：

$$r(t) = U(t)\mathrm{e}^{(\sigma z(t))}$$

如果 $i=0$，…，N 是第 i 个时间步，$j=1$，…，i 是树中的第 j 个空间步，能够从下式中确定节点（i，j）在 Δt 时段内的短期汇率水平.

$$r_{i,j} = U(i)\mathrm{e}^{(\sigma(i)j\sqrt{\Delta t})} \tag{A. 2}$$

最初，假设 $U(0) = r_{0,0}$. $U(i)$ 和 $\sigma(i)$ 能够通过 Arrow-Debrue 牌价确定. 由于利率的对数正态分布假设，BDT 模型对于债券价格和债券期权没有闭型的解析解. BDT 树构建过程的细节，请参照 Black、Derman 和 Toy(1990)，Jamshidian(1991)，以及 Clewlow 和 Strickland (1999). BDT 树的 C++实现，请参照 London(2004) .

在 Matlab 中构建 BDT 树，调用函数 bdttree，BDTTree = bdttree (VolSpec，RateSpec，TimeSpec)，在一棵重组树上创建了一个包含时间和利率信息的结构，参数介绍如表 A. 1 所示.

表　A. 1

参　数	描　述
VolSpec	波动性过程详述. 参照 bdtvolspec 获取关于波动性过程的信息
RateSpec	初始汇率曲线中的利率详述. 参照 intenvset 获取声明一个利率变量的信息
TimeSpec	树的时间显示说明. 定义 BDT 树的观察日期和日期对于时间映射与价格产生公式的组合规则. 参照 bdttimespec 获取关于树的结构的信息

下面的 Matlab 代码根据特定的 VolSpec、RateSpec 和 TimeSpec 构建了一棵 BDT 树.

```
load deriv.mat

Compounding = 1;
ValuationDate = '01-01-2000';
StartDate = ValuationDate;
EndDates = ['01-01-2001'; '01-01-2002'; '01-01-2003';
'01-01-2004'; '01-01-2005'];
Rates = [.05;.0575;.0625;.0675;.07];
Volatility = [.1;.095;.09;.085;.08];

RateSpec = intenvset('Compounding', Compounding,...
                     'ValuationDate', ValuationDate,...
                     'StartDates', StartDate,...
                     'EndDates', EndDates,...
                     'Rates', Rates);

BDTTimeSpec = bdttimespec(ValuationDate, EndDates, Compounding);
BDTVolSpec = bdtvolspec(ValuationDate, EndDates, Volatility);
BDTTree = bdttree(BDTVolSpec, RateSpec, BDTTimeSpec);
```

假设为一个 cap 定价，使用的 BDT 树参数如下：

```
CapStrike = 0.065;
Settle = '01-Jan-2000';
Maturity = '01-Jan-2004';
CapReset = 1;
Principal = 100000;

We call the function

Price = capbybdt(BDTTree, CapStrike, Settle, Maturity, Principal)

Price =

    3.3034
```

假设在 Matlab 中为债券期权定价，调用函数如下：

```
[Price, PriceTree] = optbndbybdt(BDTTree, OptSpec, Strike,...
     ExerciseDates, AmericanOpt, CouponRate, Settle, Maturity,...
       Period, Basis, EndMonthRule, IssueDate, FirstCouponDate,...
       LastCouponDate, StartDate, Face, Options)
```

参数介绍见表 A. 2.

表 A. 2

参 数	描 述
BDTTree	由 BDT 树创建的远期汇率树结构
OptSpec	‘看涨’或‘看跌’的金融工具数量 (NINST) ×1 的字符串单元数组
Strike	欧式期权：NINST×1 的执行价格向量 百慕大期权：NINST×敲定的价格数（NSTRIKES）的执行价格矩阵 每一行是一个期权的列表，如果一个期权的执行机会（exercise opportunities）少于敲定价格数量，这一行的结尾用空值 NaN 填充 美式期权：代表每种期权执行价格值的 NINST×1 向量

（续）

参　数	描　述
ExerciseDates	NINST×1（欧式期权）或者 NINST×NSTRIKES（百慕大期权）的执行日期矩阵．每一行是一种期权的时间表．对于欧式期权，只有一个执行日期：此期权的到期日．对于美式期权，NINST×2 执行日期范围的向量．对于每一个金融工具，期权可以在那一行的日期对之间或包含这两个日期的票据日期执行．如果只有一个非空的日期，或者如果 ExerciseDate 是 NINST×1，期权可以在标的债券 Settle 和列出的单个执行日之间执行
AmericanOpt	NINST×1 的标识向量：0（欧式期权或百慕大期权）或 1（美式期权）
CouponRate	十进制年率
Settle	结算日期．一系列的日期数字或日期字符串向量．Settle 必须早于或等于
Maturity	到期日．一系列的日期数字或日期字符串向量
Period	（可选）债券每年的息票，整数向量，可取的值是：1，2，3，4，6 和 12．默认值是 2
Basis	（可选）金融工具的基础结算日．一个整数向量．0＝实际值/实际值（默认），1＝30/360（SIA），2＝实际值/360，3＝实际值/360，4＝30/360（PSA），5＝30/360（ISDA），6＝30/360（欧洲的），7＝实际值/365（日本的）
EndMonthRule	（可选）"月终"规则．一个向量．这个规则只有在到期日是一个月的月末才适用，并且这个月的天数要小于等于 30．0＝忽略规则，意味着债券的付息日总是这个月的相同数字代表的这一天．1＝使用规则（默认），意味着债券的息票支付日总是这个月的实际的最后一天
IssueDate	（可选）债券发行的日期
FirstCoupon-Date	债券第一次息票支付日．如果同时定义了 FirstCouponDate 和 LastCouponDate，FirstCouponDate 优先确定债券息票支付结构
LastCoupon-Date	债券到期日之前最后一个息票支付日．在没有定义 FirstCouponDate 的情况下，LastCouponDate 确定债券的息票结构．债券的息票结构不管落在哪里都在 LastCouponDate 处被截断，后面只有债券的现金流到期日
StartDate	忽略
Face	票面价值，默认值是 100
Options	（可选）与 deriveset 一起创建的衍生品定价选项结构

每一个债券的结算日设成 BDT 树的 ValuationDate．忽略债券的参数 Settle．

假设我们想要评价一个欧式债券 4%的附息票债券的看涨期权，面值的执行价格为 85，2 年期，估值日期是 2002 年 1 月 1 日；价格＝固定收益工具．我们把基本债券加入到投资组合里：

```
InstSet = instadd('Cap',CapStrike,Settle,Maturity);InstSet =
instadd(InstSet, 'Bond',0.04,'01-01-2000','01-01-2002');
InstSet = instadd(InstSet, 'OptBond',1,'Call',85,'01-01-2002',0);
instdisp(BDTSubSet)
Index Type CouponRate Settle          Maturity      Period    Basis
1     Bond 0.04       01-Jan-2000     01-Jan-2002   2         0
EndMonthRule IssueDate FirstCouponDate LastCouponDate StartDate
1            NaN           NaN             NaN            NaN
  Face
  100
Index Type Strike Settle          Maturity      CapReset Basis
2     Cap  0.06   01-Jan-2000     01-Jan-2004   1        0
  Principal
  100
```

```
Index  Type     UnderInd OptSpec Strike ExerciseDates AmericanOpt
3      OptBond  1        Call    85     01-Jan-2002   0
```

要计算风险度量，调用如下函数：

```
    [Delta Gamma Vega Price] = bdtsens(BDTTree, BDTSubSet)

Delta =

-177.8993
 207.9652
 -25.3722


Gamma =

  1.0e+003 *

    0.5004
    4.5744
    0.0720

Vega =

  -0.0002
   2.8968
  -0.0000

Price =

  96.9116
   3.3034
  13.4131
```

Delta 和 Gamma 是基于 100 基点的收益变化计算的. 所有的敏感性都返回美元敏感性. 要找到每单位美元的敏感性，我们把风险度量按照它们各自的金融工具价格分解如下：

```
-1.8357
 2.1459
-0.2618
```

A.2 Matlab 中的 Hull-White 树

Hull 和 White(1994)[2] 开发了一个通用的构建树的程序来构建符合收益率曲线的三叉树. 这个模型是 Vasick 模型[3] 的扩展：

$$\mathrm{d}r = a(\bar{r}(t) - r)\mathrm{d}t + a\mathrm{d}z$$

更普遍地表达形式如下：

$$\mathrm{d}r = (\theta(t) - ar)\mathrm{d}t + \sigma\mathrm{d}z \tag{A.3}$$

其中，$\theta(t)$ 是树拟合参数，α 是利率的平均回复速度.

为了构建 BDT 树，我们使用等距离的时间步长和一个转换概率固定于 1/2 的二项式过程，只允许调整空间步长. 在三叉树中，我们固定时间和空间步长，但是可以自由选择概率，以确保短期利率随着时间区间 Δt 的变化的分布有合适的均值和标准差以与被近似的短期利率过程匹配.

构建 Hull-White 树是一个两阶段的过程. Hull 和 White 在程序的第一阶段中构建一个三叉树来近似由 Ornstein-Uhlenbeck 过程模拟的短期利率：

$$dx = -ax\,dt + \sigma dz, \quad x(0) = 0$$

对于每个 t，$x(t)$ 关于 0 对称分布. 并且，对于小的时间变化 Δt，我们忽略 Δt 的二阶或更高阶的形式，则 x 的变化符合有限均值和方差的正态分布. 也就是说，$dx(t)=x(t+\Delta t) - x(t) \sim N(-ax(t)\Delta t, \sigma^2\Delta t)$. Δr 用来计算状态空间的利率之间的间隔，被设为：

$$\Delta r = \sigma\sqrt{3\Delta t}$$

其中，Δt 是时间步长的长度.[4] 树构建过程的下一个阶段是将 x 树上的点用 $\alpha(t)$ 来代替，因此将 x 树转化为拟合初始期限结构的 r 树. 也就是说，r 树每个节点都包含短期利率 $r(t)=x(t)+\alpha(t)$，而且 $\alpha(t)=r(t)-x(t)$. 回忆一下模型 $dr=(\theta(t)-ar)\,dt+\sigma dz$ 和 $dx=-axdt+\sigma dz$，从这些模型中可以看出，$\alpha(t)$ 必须遵循以下公式：

$$d\alpha(t) = (\theta(t) - a\alpha(t))dt$$

有关 Hull-White 树构建过程的细节，请参照 Hull-White(1994) 和 Hull(1997)、Clewlow 和 Strickland(1998). Hull-White 树（单因素和多因素模型）在 C++ 中的实现，请参照 London(2004).

在 Matlab 中构建 Hull-White 树，需要用 HWVolSpec 定义一个 Hull-White 波动性规格，输入参数是估值日、波动日期、波动曲线、α 日期和 α 曲线. 模型参数按照波动性和 α 输入值校准.

```
load deriv.mat

Compounding = 1;
ValuationDate = '01-01-2004';
StartDate = ValuationDate;
VolDates = ['12-31-2004'; '12-31-2005'; '12-31-2006'; '12-31-2007'];
VolCurve = 0.01;
AlphaDates = '01-01-2008';
AlphaCurve = 0.1;
Rates = [0.0275; 0.0312; 0.0363; 0.0415];

BKVolSpec = bkvolspec(ValuationDate, VolDates, VolCurve,... AlphaDates,
AlphaCurve);

RateSpec = intenvset('Compounding', Compounding,...
  'ValuationDate', ValuationDate,...
  'StartDates', ValuationDate,...
  'EndDates', VolDates,...
  'Rates', Rates);

HWTimeSpec = hwtimespec(ValuationDate, VolDates, Compounding);
HWTree = hwtree(HWVolSpec, RateSpec, HWTimeSpec)
```

构建了 Hull-White 树之后，我们可以使用 Matlab 函数 floorbyhw 对底价进行估计：

```
Strike = 0.03;
Principal = 100000;
Settle = '01-Jan-2005';
Maturity = '01-Jan-2009';

Price = floorbyhw(HWTree, Strike, Settle, Maturity,1,0,Principal)

Price =

  461.59
```

可以发现用 Matlab 为固定收益衍生品定价比用 C++ 编码要简单得多（把以上的代码与下

面的 C++代码比较). 假设我们要在 C++中用 Black 的 1976 模型为一种资本定价，然后校准 Hull-White 模型到利率曲线和利率上限波动性. 首先定义 Black 的类:

BlackModel01.h

```
#ifndef _BLACKMODEL__
#define _BLACKMODEL__

#include <vector>
#include <math.h>
#include "StatUtility.h"

class BlackModel
{
        public:

                BlackModel() {};
                virtual ~BlackModel() {};
                std::vector<double> priceBlackCap(std::vector<double>
                  capVol, std::vector<double> PDB, std::vector<double>
                  maturity, double Rcap, double L, double tenor);
                double BlacksFormula(double f, double P, double L,
                  double Rcap,double vol, double tau, double dtau);
};

#endif
```

方法定义如下:

BlackModel02.cpp

```
#include "BlackModel.h"

std::vector<double> BlackModel::priceBlackCap(std::vector<double> capVol,
  std::vector<double> PDB, std::vector<double> maturity, double Rcap,
  double L, double tenor)
{
        int i;
        std::vector<double> f;     // forward rates
        std::vector<double> R;     // yield price
        std::vector<double> capV;
        std::vector<double> P;
        std::vector<double> t;
        std::vector<double> caplet;
        double cap = 0.0;
        double faceValue = 0.0;
        double Ps = 0.0;
        double tmp = 0.0;

        std::vector<double>::iterator iter;
        faceValue = L*(1 + Rcap*tenor);

        for (iter = capVol.begin(); iter != capVol.end(); iter++)
        {
                tmp = *iter;
                capV.push_back(tmp);
        }
        for (iter = PDB.begin(); iter != PDB.end(); iter++)
        {
                tmp = *iter;
```

```
            P.push_back(tmp);
        }
        for (iter = maturity.begin(); iter != maturity.end(); iter++)
        {
                tmp = *iter;
                t.push_back(tmp);
        }
        for (i = 0; i < capVol.size(); i++)
        {
                tmp = -(1/t[i])*(log(P[i]));
                R.push_back(tmp);
                tmp = -(1/tenor)*log(P[i+1]/P[i]);
                f.push_back(tmp);
        }

        for (i = 0; i < capVol.size()-1; i++)
        {
                tmp = BlacksFormula(f[i],P[i],L,Rcap,capV[i],t[i],tenor);
                caplet.push_back(tmp);
        }

        return caplet;
}

double BlackModel::BlacksFormula(double f, double P, double L, double Rcap,
  double vol, double tau, double dtau)
{
        StatUtility util;
        double d1 = (log(f/Rcap) + ((vol*vol)/2)*tau)/(vol*sqrt(tau));
        double d2 = d1 - vol*sqrt(tau);

        return P*dtau*L*(f*util.normalCalc(d1) - Rcap*util.normalCalc(d2));
}
```

我们在 HullWhite 类里定义资产定价函数，称作 BlackModel 类里的 priceBlackCap 方法：

HullWhite_priceCapHW.cpp

```
#include "HullWhite.h"
#include <numeric>

double HullWhite::priceCapHW(vector<double> mats, vector<double> vols,
  vector<double> rates, double a, double FV, double vol, double Rcap,
  double tenor, double L)
{

        int i;
        TNT::Array2D<double> B(NUM,NUM);
        double d1, d2;
        double K = L;
        double volP, a1;
        double tmp = 0.0;
        double totalsum = 0.0;
        double sum = 0.0;
        double sum1 = 0.0;
        double sum2 = 0.0;
        double tau, tau1, tau2, tau3;
        double epsilon = 0.001;
        double error = 0.0;
        double error1 = 0.0;
```

```
        double volSum2 = 0.0;
        double alpha1, alpha2 = 0;
        double d1prime, d2prime, d3prime, d4prime;
        double volPrime = 0.0, a2;
        double aprime = 0.0;
        double aPrime = 0.0;

        double diff = 0.0;
        double SSE = 0.0;
        double val = 0.0;
        vector<double> P;
        vector<double> model;
        vector<double> market;
        int cnt = 0;
        int len = vols.size();

        std::cout << "Hull White Cap Calibration" << endl << endl;
        std::cout << "Cap Rate :  " <<  Rcap << endl;
        std::cout << "Principal:  " << L << endl;
        std::cout << "Face Value Bond: " << FV << endl;
        std::cout << "Tenor: " << tenor << endl;

        // compute pure discount bond prices
        for (i = 0; i < len; i++)
        {
                tmp = exp(-mats[i]*rates[i]);
                P.push_back(tmp);
        }

        // compute market quotes
        BlackModel bm;
        market = bm.priceBlackCap(vols,P,mats,Rcap,L,tenor);
        val = std::accumulate(market.begin(),market.end(),0.0);

        for (i = 0; i < len-1; i++)
        {
           volP = sqrt(((vol*vol)/(2*a*a*a))*(1 - exp(-2*mats[i]*a))*(1 -
             exp(-a*(mats[i+1]-mats[i])))*(1 - exp(-a*(mats[i+1]-
             mats[i]))));
           d1 = log((FV*P[i+1])/(K*P[i]))/volP + volP/2;
           d2 = d1 - volP;
           tmp = K*P[i]*util.normalCalc(-d2) - FV*P[i+1]*
             util.normalCalc(-d1);
           model.push_back(tmp);
        }
        sum = 0;
        SSE = 0;
        for (i = 0; i < len-1; i++)
        {
                sum = sum + ((model[i] - market[i])/market[i])*((model[i] -
                  market[i])/market[i]);
                SSE = SSE + (model[i] - market[i])*(model[i] - market[i]);
        }

        // Minimization routine
        totalsum = sum;
        alpha1 = vol;
        a1 = a;
        do
        {
```

```
        sum1 = 0;
        sum2 = 0;
        volSum2 = 0;
        for (i = 0; i < len-2; i++)

        {
          tau = mats[i+1] - mats[i];
          tau1 = mats[i];
          tau2 = mats[i+1];
          tau3 = mats[i+1] + mats[i];
          volP = sqrt(((alpha1*alpha1)/(2*a1*a1*a1))*(1 -
            exp(-2*tau1*a1))*(1 - exp(-a1*(tau)))*(1 - exp(-a1*(tau))));
          // compute d1 and d2
          d1 = log((FV*P[i+1])/(K*P[i]))/volP + volP/2;
          d2 = d1 - volP;
          aprime = -3*pow(a1,-4);
          volPrime = 0.5*(pow(volP,-0.5))*(2*alpha1/(2*a1*a1*a1))*((1 -
            exp(-2*tau1*a1))*(1 - exp(-a1*(tau)))*(1 - exp(-a1*(tau))));
          aPrime = 0.5*pow(volP,-0.5)*((0.5*alpha1*alpha1*pow(a1,-3)*
            (2*tau*exp(-a1*tau) - 2*tau*exp(-2*a1*tau) +
            2*tau1*exp(-2*tau1*a1) - 2*tau3*exp(-a1*tau3) +
            2*tau2*exp(-2*a*tau2)) +(0.5*aprime*alpha1*alpha1)*
            (1 - 2*exp(-a1*tau) + exp(-2*a1*tau)
             - exp(-2*a1*tau1) + 2*exp(-a1*tau3) - exp(-2*a1*tau2))));
          d1prime =  -((log(FV*P[i+1]/K*P[i]))/(volP*volP))*volPrime +
            0.5*volPrime;
          d2prime = d1prime - volPrime;
          d3prime = -((log(FV*P[i+1]/K*P[i]))/(volP*volP))*aPrime +
            0.5*aPrime;
          d4prime = d3prime - aPrime;
          sum1 = sum1 + (model[i] - market[i])/market[i];
          volSum2 = volSum2 + (K*P[i]*util.normalCalcPrime(-d2)*
            (-d2prime) -
            FV*P[i+1]*util.normalCalcPrime(-d1)*(-d1prime))/market[i];
          sum2 = sum2 + (K*P[i]*util.normalCalcPrime(-d2)*(-d4prime) -
            FV*P[i+1]*util.normalCalcPrime(-d1)*(-d3prime))/market[i];
        }
        alpha2 = alpha1 - sum1/volSum2;
        error = alpha2 - alpha1;
        alpha1 = alpha2;

        a2 = a1 - sum1/-sum2;
        error1 = a2 - a1;
        a1 = a2;
        cnt++;
        if (cnt > 20)
          break;
    }
    while ((error > epsilon) || (error1 > epsilon));

    if (cnt < 20)
    {
      std::cout << "Calibrated alpha = " << a1  << endl;
      std::cout << "Calibrated vol = " << alpha1 << endl;
    }
    else
          std::cout << "No Convergence for Calibration.  Try
            different values." << endl;

    return val;
}
```

我们从 HullWhite 文件夹里读取一个包含 caplet 到期日、波动性和收益的叫做 Cap. txt 文件. 主函数如下：

HullWhite_main. cpp

```
#include <fstream>
#include <iostream>
#include <strstream>
#include "HullWhite.h"
#include "BDT.h"

void main()
{
        char buffer[SIZE_X];
        char dataBuffer[SIZE_X];
        char* str = NULL;
        const char* file = "c:\\Caplet.txt";
        vector<double> m;
        vector<double> v;
        vector<double> r;
        double mat = 0.0;
        double rate = 0.0;
        double a = 0.10;            // mean speed of reversion
        double vol = 0.10;          // volatility
        double strike = 0.055;      // exercise price
        double T = 5;               // option maturity
        double coupon = 0.0;        // bond coupon rate
        double swapMat = 5;         // swap Maturity
        double swaptionMat = 3;     // swaption maturity
        double price = 0.0;         // price of bond
        double faceValue = 106.7; // face value of bond
        double principal = 100;     // principal amount of bond

    HullWhite hw;
        BDT bdt;

        // read from file
        ifstream fin;               // input file stream
        fin.clear();
        fin.open(file);

        if (fin.good())
        {
                while (!fin.eof())
                {
                        fin.getline(buffer,sizeof(buffer)/
                          sizeof(buffer[0]));
                        istrstream str(buffer);

                        // Get data
                        str >> dataBuffer;
                        mat = atof(dataBuffer);
                        m.push_back(mat);

                        str >> dataBuffer;
                        vol = atof(dataBuffer);
                        v.push_back(vol);
                        str >> dataBuffer;
                        rate = atof(dataBuffer);
                        r.push_back(rate);
```

```
            }
    }
    else
      std::cout << "File not good!" << "\n";

     fin.close();

     price = hw.priceCapHW(m,v,r,a,faceValue,vol,0.07,0.25,principal);
     std::cout << "Cap price = " << price << endl << endl;

     vector<double> zero;
     zero.push_back(0.055);
     zero.push_back(0.0575);
     zero.push_back(0.0600);
     zero.push_back(0.0625);
     zero.push_back(0.0650);
     zero.push_back(0.0675);
     zero.push_back(0.07);
     zero.push_back(0.0710);
     zero.push_back(0.0740);
     zero.push_back(0.0775);

     vector<double> vols;
     vols.push_back(0.06);
     vols.push_back(0.065);
     vols.push_back(0.07);
     vols.push_back(0.075);
     vols.push_back(0.08);
     vols.push_back(0.085);
     vols.push_back(0.09);
     vols.push_back(0.095);
     vols.push_back(0.10);
     vols.push_back(0.105);

     price = hw.priceDiscountBondOptionsHW(zero,10,2,vol,a,2,strike,
       'C','E');
     std::cout << "HW Discount Bond Option Price = " << price << endl;

     price = bdt.priceDiscountBondsBDT(zero,5,5,vol,strike,'C','E',
       zero[0]);
     std::cout << "BDT Discount Bond Option Price = " << price <<
       endl << endl;

     price = hw.payerSwaptionHW(zero,swapMat,swaptionMat,vol,a,strike,
       'E',coupon);
     std::cout << "HW European pay swaption price = " << price << endl;

     price = hw.payerSwaptionHW(zero,swapMat,swaptionMat,vol,a,strike,
       'A',coupon);
     std::cout << "HW American pay swaption price = " << price <<
       endl << endl;
       price = hw.receiverSwaptionHW(zero,swapMat,swaptionMat,vol,a,
         strike,'E',coupon);
       std::cout << "HW European recevier swaption price = " << price <<
         endl;
       price = hw.receiverSwaptionHW(zero,swapMat,swaptionMat,vol,a,
         strike,'A',coupon);
       std::cout << "HW American recevier swaption price = " << price <<
         endl;
}
```

输入出如下：

```
Hull White Cap Calibration

Cap Rate: 0.07
Principal: 100
Face Value Bond: 100
Tenor: 0.25
Calibrated alpha = -0.317789
Calibrated vol = 0.0984802
Cap price = 3.48947
```

这个 cap 价格是每 100 美元本金 0.034 8 美元，所以 cap 在 1 000 000 美元 7%的利率下是 34 800 美元 .

A. 3 Matlab 中的 Black-Karasinski 树

我们可以将 Hull-White 树构建程序扩充到短期利率的对数正态过程，短期利率不允许产生负利率. 因此，它被业界广泛使用. 对数正态 HW 模型或受限制的 Black-Karasinski (1991)[5] 模型可以表述为：

$$\mathrm{d}\ln r(t) = a(\ln \bar{r} - \ln r(t))\mathrm{d}t + \sigma \mathrm{d}z \tag{A. 4}$$

或者写成更一般的 Black－Karasinski 形式

$$\mathrm{d}\ln r(t) = (\theta(t) - a\ln r(t))\mathrm{d}t + \sigma \mathrm{d}z \tag{A. 5}$$

上式中与时间相关的项用于校准模型到最初的收益曲线. 常参数 a 和 σ 确定了波动性期限结构. 从式(A. 5) 中，使用 Ito 推论，得到：

$$\mathrm{d}r(t) = r(t)\left[\theta(t) + \frac{\sigma^2}{2} - a\ln r(t)\right]\mathrm{d}t + \sigma\, r(t)\mathrm{d}z(t)$$

我们可以将 Black-Karasinski 模型应用到 A. 2 节中介绍的两阶段方法来构建短期利率树(用来计算没有闭环分析解的贴现债券和债券期权). 首先，设 $x=\ln r$，产生如下过程：

$$\mathrm{d}x = (\theta(t) - ax)\mathrm{d}t + \sigma \mathrm{d}z$$

然后设 $\theta(t)=0$，过程变为与之前一样：

$$\mathrm{d}x = - ax\mathrm{d}t + \sigma \mathrm{d}z$$

在 0 时刻，$x=0$. 我们与之前一样假设 $\Delta r=\Delta x=\sigma\sqrt{3\Delta t}$. 树以等距离的状态空间和时间步长对称. 然而，我们可以修正模型（包括 Hull-White 模型）使其随时间变化，在 Δt 时间内变化 Δr，也就是说，对于每个 i，$\Delta t_i=t_{i+1}-t_i$，$\Delta r_i=\sigma\sqrt{3\Delta t_i}$. 在节点（$i$，$j$），计算 $x_{i,j}=j\Delta x$. 然后计算出每个节点的 Arrow-Debreu 债券价格和 $i=1$，…，N 的每一个时间步长的偏移参数 α_i. 选择 α_i 来正确定价一个 $(i+1)\Delta t$ 的到期贴现债券. 节点 j 处的 Δt 期间利率在 $i\Delta t$ 时刻变为：

$$r_{i,j} = \exp(\alpha_i + x_{i,j}) = \exp(\alpha_i + j\Delta x) \tag{A. 6}$$

对于 Black-Karasinski 树构建的细节，参见 Hull(1997) 和 Clewlow 和 Strickland(1998).

为在 Matlab 中构建 Black-Karsinski 树，需要用 bktree 产生一个 BKTree，参数包括 BKVolSpec、BKTimeSpec 和 BKRateSpec：

```
Compounding = -1;
ValuationDate = '01-01-2004';
StartDate = ValuationDate;
VolDates = ['12-31-2004'; '12-31-2005'; '12-31-2006'; '12-31-2007'];
```

```
VolCurve = 0.01;
AlphaDates = '01-01-2008';
AlphaCurve = 0.1;
Rates = [0.0275; 0.0312; 0.0363; 0.0415];

BKVolSpec = bkvolspec(ValuationDate, VolDates, VolCurve,...
  AlphaDates, AlphaCurve);          RateSpec = intenvset
  ('Compounding', Compounding,...
   'ValuationDate', ValuationDate,...
   'StartDates', ValuationDate,...
   'EndDates', VolDates,...
   'Rates', Rates);
BKTimeSpec = bktimespec(ValuationDate, VolDates, Compounding);
  BKTree = bktree(BKVolSpec, RateSpec, BKTimeSpec)

BKTree =

      FinObj: 'BKFwdTree'
      VolSpec: [1x1 struct]
      TimeSpec: [1x1 struct]
      RateSpec: [1x1 struct]
      tObs: [0 0.9973 1.9973 2.9973]
      dObs: [731947 732312 732677 733042]
      CFlowT: {[4x1 double]  [3x1 double]  [2x1 double]  [3.9973]}
      Probs: {[3x1 double]  [3x3 double]  [3x5 double]}
      Connect: {[2]  [2 3 4]  [2 2 3 4 4]}
      FwdTree: {1x4 cell}
```

要查看树，键入 treeviewer(BKTree). Matlab 中树显示如图 A.1 所示. 一旦构建了树，我们可以像为债券和资本定价那样为固定收益债券定价.

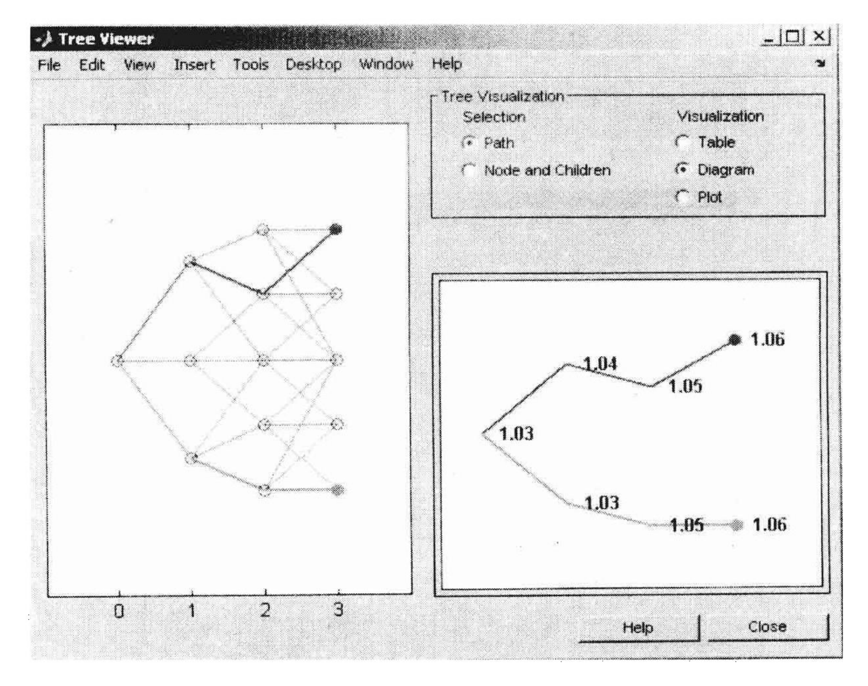


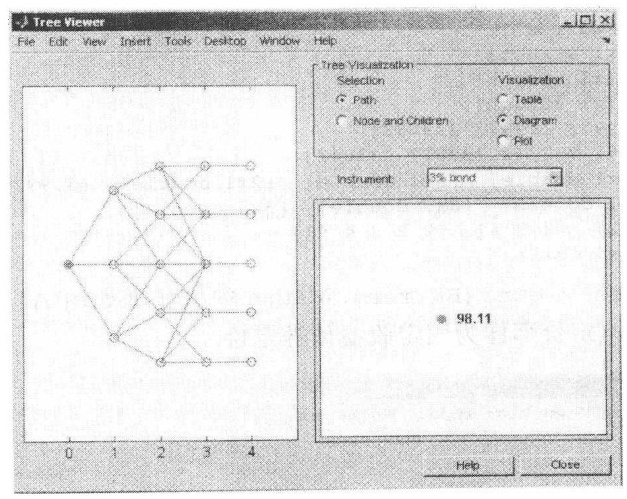
图 A.1 Black-Karasinski 树

资料来源：Mathworks，www.mathworks.com/products/derivatives/functionlist.html.

```
load deriv;
BKSubSet = instselect(BKInstSet,'Type', {'Bond', 'Cap'});
instdisp(BKSubSet)
Index Type   CouponRate Settle      Maturity    Period     Name ...
1     Bond   0.03       01-Jan-2004 01-Jan-2007 1          3% bond
2     Bond   0.03       01-Jan-2004 01-Jan-2008 2          3% bond
Index Type   Strike     Settle      Maturity    CapReset...Name ...
3     Cap    0.04       01-Jan-2004 01-Jan-2008 1          4% Cap
[Price, PriceTree] = bkprice(BKTree, BKSubSet);

Price =
  98.1096
  95.6734
  2.2706
```

要查看树，键入 treeviewer(PriceTree,BKSubSet)，即可在如图 A.2 中查看金融工具的价格.



first 3% bond

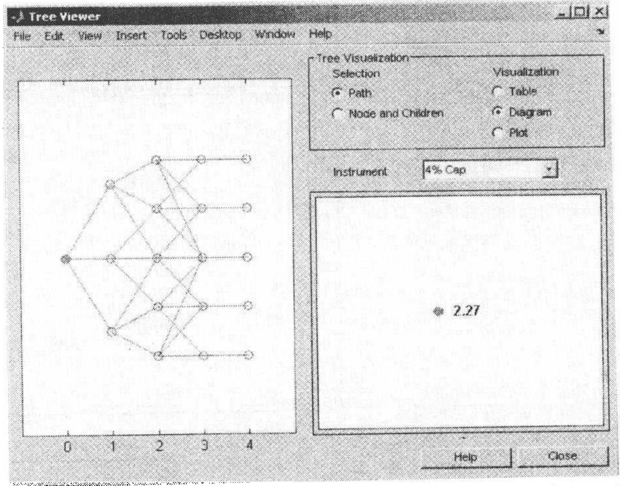


图 A.2 用 Black-Karasinski 树为 first 3% bond 和 4% cap 定价

资料来源：Mathworks，www.mathworks.com/products/deriveatives/functionlist.html.

A. 4 Matlab 中的 HJM 定价

Heath、Jarrow 和 Morton（HJM）（1992）[6] 为模拟利率（远期利率而非即期利率）开发了一个总体框架. 在 HJM 框架中，波动率函数是整个远期利率曲线的函数. HJM 假设在一个客观测度下，$f(t, T)$ 的演化，瞬时远期利率由下式给出：

$$df(t,T) = \alpha(t,T)dt + \sum_{i=1}^{N}\sigma_i(t,Tf(t,T))dz_i(t) \tag{A. 7}$$

波动率函数 $\sigma_i(\cdot)$ 依赖于整个远期利率曲线，彻底说明了这个远期利率模型. 远期利率的波动性和相关性都由波动率函数决定.

关于单因素和双因素的 HJM 离散状态模型（包括模型上的无套利限制），参见 Jarrow (2002). 在 Matlab 中构建 HJM 树的代码实现直接基于这些离散状态模型. 这些 HJM 离散状态模型的 C++ 实现，参见 London(2004).

要在 Matlab 中用 HJM 树为固定收益债券定价，首先需要用 `hjmtree` 函数构建一个 HJM 树. 这个函数有三个参数：波动性说明（`VolSpec`）、利率期限结构说明（`RateSpec`）和时间说明（`TimeSpec`）.

`VolSpec` 是说明远期利率波动过程的结构. 你可以用两个函数中的任意一个创建 `VolSpec`:`hjmvolspec` 或 `bdtvolspec`.`hjmvolspec` 函数支持多达三个因素的说明. 它处理这些模型的利率期限结构的波动性：

- **Constant volatility (Ho-Lee):** $\sigma(t,T)$ ==Sigma_0
  `VolSpec = hjmvolspec('Constant', Sigma_0)`
- **Stationary volatility:** $\sigma(t,T)$ =Vol(T-t) = Vol(Term)
  `VolSpec = hjmvolspec('Stationary', CurveVol, CurveTerm)`
- **Exponential volatility:** $\sigma(t,T)$ =Sigma_0*exp(-Lambda*(T-t))
  `VolSpec = hjmvolspec('Exponential', Sigma_0, Lambda)`
- **Vasicek, Hull-White:** $\sigma(t,T)$ =Sigma_0*exp(-Decay(T-t))
  `VolSpec = hjmvolspec('Vasicek', Sigma_0, CurveDecay, CurveTerm)`
- **Nearly proportional stationary:** $\sigma(t,T)$ =Prop(T-t)* max(SpotRate(t), MaxSpot)
  `VolSpec = hjmvolspec('Proportional', CurveProp, CurveTerm, MaxSpot)`

单因素模型假设利率期限结构受单一的不确定性资源影响. 融入多个因素可以让你规定利率期限结构在形状和位置上的不同类型的偏移. 函数 `hjmvolspec` 产生结构 `VolSpec`，规定了波动过程 $\sigma(t, T)$，此过程在远期利率树的构建过程中使用过. 这里，T 表示远期利率的开始时间，t 代表观测期. 波动性过程可以从调用函数创建模型中依次说明的多个因素的组合来构建. 每个因素说明以表示此因素名字的字符串开始，接着是相关的参数. 时间价值 t、T 和 Term 在由 `hjmtimespec` 的 Compounding 输入指定的息票间隔单位之间取值. 例如，如果 Compounding=2，Term=1 表示半年期（6 个月）.

`bdtvolspec` 函数只支持一个波动率因素. 波动率在树上的节点对之间保持不变. 可以用十进制的向量提供输入的波动率值.

RateSpec 是初始利率曲线上利率的详细说明. 用函数 intenvset 创建此结构.

[RateSpec,RateSpecOld]= intenvset (RateSpec,'Argument1',Value1,'Argument2',Value2,…) 函数用以创建一个利率期限结构(RateSpec)，输入参数列表具体为参数名字/参数值序列对. 序列对的参数名字部分必须是输出结构 RateSpec 的一个有效区域，将序列对的参数值部分分派到它对应的区域.

如果指定了可选参数 RateSpec,intenvset 通过改变具名变量为指定的值，依据新值重新计算参数，来修正现存的利率期限结构 RateSpec.

[RateSpec,RateSpecOld]= intenvset 将所有字段设为 [] 来创建新的利率期限结构. 没有输入和输出参数的 intenvset 展示了一个参数名字和可能取值的列表. RateSpecOld 是一个包含一个利率期限结构的属性在表 A.3 介绍的变化之前的一个结构. 参数可以从表 A.4 中选择并以任何顺序排列.

仅打出唯一确定参数的几个主要字母即可. 参数名字的大小写忽略. 当创建一个新的 RateSpec，传递到 intenvset 的参数集合必须包括 StartDates、EndDates，以及 Rates 或 Disc 两者中的一个. 调用没有输入或输出参数的 intenvset 来展示参数名字和可能值的列表. TimeSpec 是树的时间分布说明. 使用函数 hjmtimespec 或 bdttimespec 来创建这个变量. 它表示利率报价的级别日期和级别时间之间的映射. 这个结构间接确定了树中的级别数目. hjmtimespec 函数详细说明了 HJM 利率树的时间结构.

表 A.3

RateSpec	(可选) 待改变的现存利率结构说明，很可能创建于对函数 intenvset 的前一次调用

表 A.4

Compounding	标量值，表示输入的零息利率按年计算时以复利计算. 默认值=2. 这个参数决定贴现因子的计算公式：Compunding=1，2，3，4，6，12. Disc= (1+ Z/F)^(- T),F 是复合利率，Z 是零息利率，T 是时间期间单位. 例如，T=F 是一年. Compounding=365. Disc= (1+ Z/F)^(- T),F 是基本年的天数，T 是按基期计算已经过去的天数. Compounding= - 1. Disc= exp(- T* Z),T 是按年的时间
Disc	投资期间的点数(NPOINTS)×曲线数(NCURVES)单位债券价格矩阵，始于现金流估值的时间 StartDates，止于现金流收到的时间 EndDates
Rates	点数(NPOINTS)×曲线数(NCURVES)的十进制表示的收益率矩阵. 例如，5%是 0.05. 收益率是投资期间的收益，始于现金流估值的时间 StartDates，止于现金流收到的时间 EndDates
EndDates	NPOINTS×1 向量或一系列到期日的标量，结束贴现期的时间间隔
StartDates	NPOINTS×1 向量或一系列日期的标量，启动开始贴现期的时间间隔. 默认值=Valuation Date
Valuation-Date	(可选) 序列日期形式的标量值，代表以 StartDates 和 EndDates 输入的投资期间的观察日期. 默认值=min(Start Dates)
Basis	(可选) 金融工具以天计算的基期. 整数向量. 0=实际/实际 (默认值)，1=30/360 (SIA 华盛顿协议)，2=实际/360，3=实际/365，4=30/360 (PSA)，5=30/360 (ISDA)，6=30/360 (欧洲)，7=实际/365 (日本)
EndMonthRule	(可选) 月份结束规则. 一个向量. 此规则只在到期日是一个 30 天或更少天数的月份的月末时适用. 0=忽略规则，意味着债券的息票支付日期总是在一个月的同一个数值天. 1=启用规则 (默认)，表明债券的息票支付日期总是在一个月的最后一天

描述

TimeSpec= hjmtimespec 为一个 HJM 树设定级数和点次数，并且确定利率报价的日期和时间的映射.

TimeSpec 是规定 hjmtree 的时间布局的结构. 观察日期是[Settle;Maturity(1:end-1)]. 因为最后一个观测存储了一个远期利率，此树可以将现金流估值到到期日.

语法

```
TimeSpec = hjmtimespec(ValuationDate, Maturity, Compounding)
```

参数

表 A.5

ValuationDate	标量日期标明树中的定价日期和第一个观测日期. 表示为序列日期数字或者日期字符串
Maturity	树的级数（或深度）. 级数(NLEVELS)×1 的日期向量标明树中的现金流日期. 这些到期日的现金流落在树节点上. 到期日应该按升序排列
Compounding	（可选）标量值，表示输入的零息利率按年计算时以复利计算. 默认值=1. 这个参数决定贴现因子的计算公式：Compunding=1，2，3，4，6，12. Disc= (1+ Z/F)^(- T),F 是复合利率，Z 是零息利率，T 是时间期间单位. 例如，T=F 是一年. Compounding= 365. Disc= (1+ Z/F)^(- T),F 是基本年的天数，T 是按基期计算已经过去的天数. Compounding= - 1. Disc= exp(- T* Z),T 是按年的时间

示例

说明一个有半年节点（每 6 个月）的 8 期树. 用指数复利来汇报利率.

```
Compounding = -1;
ValuationDate = '15-Jan-1999';
Maturity = datemnth(ValuationDate, 6*(1:8)');
TimeSpec = hjmtimespec(ValuationDate, Maturity, Compounding)

TimeSpec =

          FinObj: 'HJMTimeSpec'
   ValuationDate: 730135
        Maturity: [8x1 double]
     Compounding: -1
           Basis: 0
    EndMonthRule: 1
```

创建一个 HJM 波动率和定价模型

HJM 波动率说明示例

考虑一个使用单因子（具体地，一个常量 sigma 因子）的模型. 这个常数因子的说明只需要一个参数：σ 值. 这种情况下，σ 值对应 0.1.

```
HJMVolSpec = hjmvolspec('Constant', 0.10)

HJMVolSpec =

FinObj: 'HJMVolSpec'
FactorModels: {'Constant'}
FactorArgs: {{1x1 cell}}
SigmaShift: 0
NumFactors: 1
NumBranch: 2
PBranch: [0.5000 0.5000]
Fact2Branch: [-1 1]
```

VolSpec 结构的 NumFactors 域，VolSpec.NumFactors=1 表明用于产生 VolSpec 的因子数是 1. FactorModels 域暗示这是一个常数因子，NumBranches 域指明分支的数量. 这样，得到的树的每个节点都有 2 个分支，一支向上，一支向下.

现在考虑由一个比例因子和指数因子构成的波动率过程.

```
% Exponential factor
Sigma_0 = 0.1;
Lambda = 1;
% Proportional factor
CurveProp = [0.11765; 0.08825; 0.06865];
CurveTerm = [ 1 ; 2 ; 3 ];
% Build VolSpec
HJMVolSpec = hjmvolspec('Proportional', CurveProp, CurveTerm,...
1e6,'Exponential', Sigma_0, Lambda)

HJMVolSpec =
FinObj: 'HJMVolSpec'
FactorModels: {'Proportional' 'Exponential'}
FactorArgs: {{1x3 cell} {1x2 cell}}
SigmaShift: 0
NumFactors: 2
NumBranch: 3
PBranch: [0.2500 0.2500 0.5000]
Fact2Branch: [2x3 double]
```

输出表明波动率说明是由 2 个因素产生的. 树的每个节点有三个分支. 从上到下每个分支的概率分别是 0.25、0.25 和 0.5.

要构建一个 HJM 树，用如下代码：

```
Compounding = -1;
ValuationDate = '01-01-2000';
StartDate = ValuationDate;
EndDates = ['01-01-2001'; '01-01-2002'; '01-01-2003';
      '01-01-2004'; '01-01-2005'];
Rates = [.1; .11; .12; .125; .13];
Volatility = [.2; .19; .18; .17; .16];
CurveTerm = [1; 2; 3; 4; 5];

HJMVolSpec = hjmvolspec('Stationary',Volatility,CurveTerm);
   RateSpec = intenvset('Compounding', Compounding,...
                  'ValuationDate', ValuationDate,...
                  'StartDates', StartDate,...
                  'EndDates', EndDates,...
                  'Rates', Rates);
HJMTimeSpec = hjmtimespec(ValuationDate, EndDates, Compounding);
HJMTree = hjmtree(HJMVolSpec, RateSpec, HJMTimeSpec);
```

我们可以用 Matlab 的函数 `treeviewer(HJMTree)` 来查看这个树，如图 A.3 所示.注意，这些分支呈指数级增长并且不再重组.由于分支数目呈指数级增长，所以它是茂密的树.

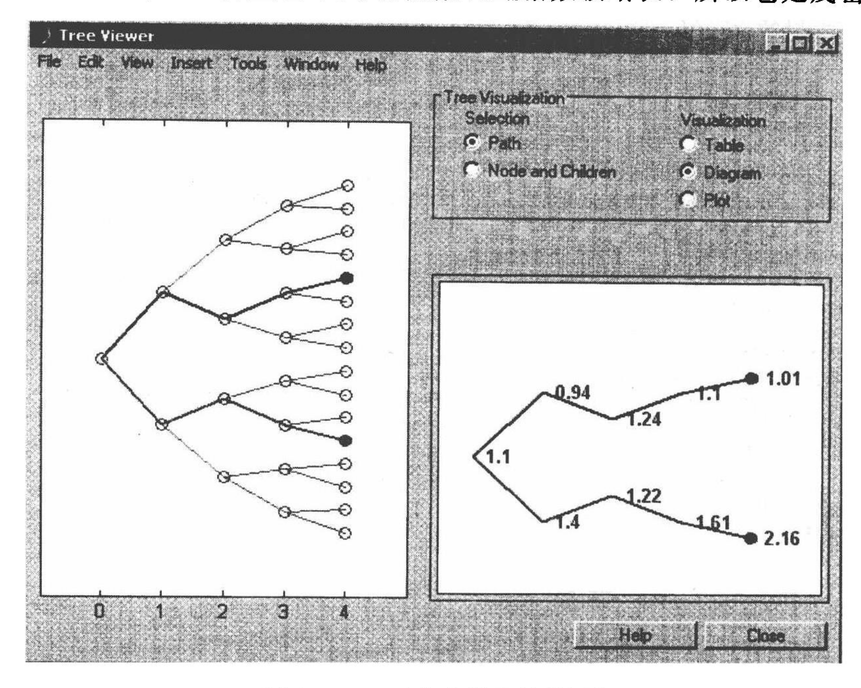


图 A.3　Matlab 声明的 HJM 树

资料来源：MathWorks，www.mathworks.com/derivatives/functionlist.html.

既然已经构建了 HJM 树，我们就可以为固定收益的金融工具定价.首先载入债券和资本数据的数据集.

```
HJMSubSet = instselect(HJMInstSet,'Type', {'Bond', 'Cap'});
instdisp(HJMSubSet)

Index Type CouponRate Settle         Maturity       Period Basis
EndMonthRule IssueDate FirstCouponDate LastCouponDate StartDate
Face Name    Quantity
1      Bond  0.04       01-Jan-2000    01-Jan-2003     1       NaN    NaN
NaN        NaN             NaN            NaN          NaN   4% bond  100
2      Bond  0.04       01-Jan-2000    01-Jan-2004     2       NaN    NaN
NaN        NaN             NaN            NaN          NaN   4% bond   50

Index Type Strike Settle         Maturity       CapReset Basis Principal
   Name  Quantity
3      Cap   0.03   01-Jan-2000    01-Jan-2004    1         NaN    NaN
   3% Cap  30

[Price, PriceTree] = hjmprice(HJMTree, HJMSubSet)

Price =

   79.3878
   73.0426
   45.1451
```

```
PriceTree =

    FinObj: 'HJMPriceTree'
     PBush: {1x6 cell}
    AIBush: {1x6 cell}
      tObs: [0 1 2 3 4 5]
Sigma_0 = 0.0076;
Lambda = 0.0154;

% Build VolSpec
HJMVolSpec = hjmvolspec('Exponential', Sigma_0, Lambda);
RateSpec = intenvset('Rates',0.02,'StartDates','01-Jan-2000','EndDates'
  '01-Jan-2004');

HJMVolSpec =

FinObj: 'HJMVolSpec'
FactorModels: {'Exponential'}
FactorArgs: {{1x2 cell}}
SigmaShift: 0
NumFactors: 1
NumBranch: 2
PBranch: [0.5000 0.5000]
Fact2Branch: [-1 1]
```

A.5 Matlab Excel Link 示例

我们可以用 Matlab Excel Link 来通过 HJM 树（或其他的利率模型）在 Excel 里为固定收益类金融工具定价，如图 A.4 所示. 假设用 Excel 作为前端为下面的金融工具定价：

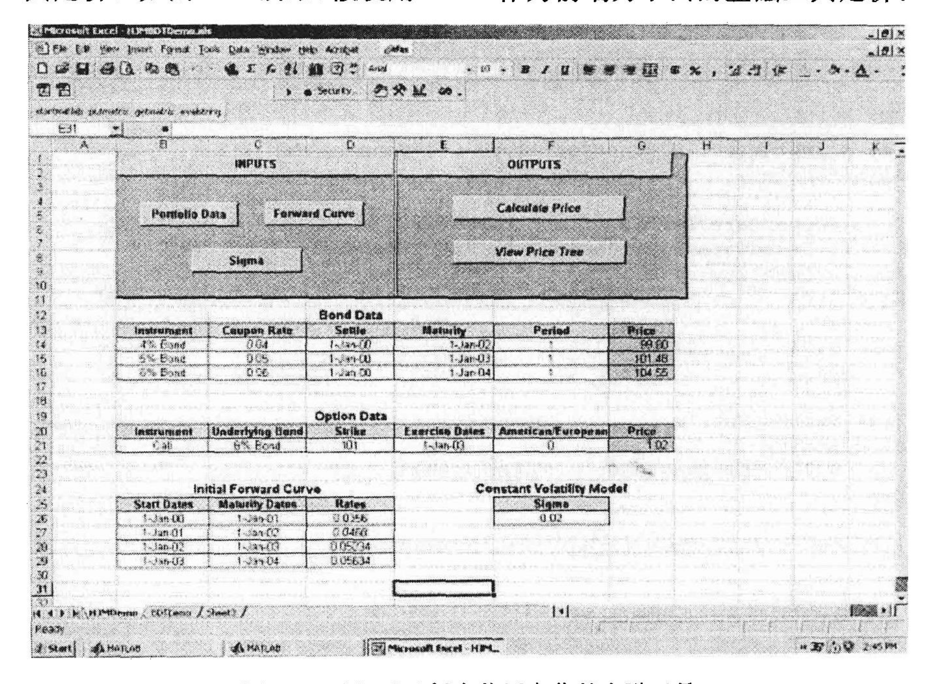


图 A.4 用 HJM 树定价固定收益金融工具

```
Index Type CouponRate Settle         Maturity       Period Basis

EndMonthRule IssueDate FirstCouponDate LastCouponDate StartDate Face Name
1      Bond 0.04        01-Jan-2000    01-Jan-2002    1       0
1      NaN        NaN            NaN            NaN       100   4% Bond
2      Bond 0.05        01-Jan-2000    01-Jan-2003    1       0
1      NaN        NaN            NaN            NaN       100   5% Bond
3      Bond 0.06        01-Jan-2000    01-Jan-2004    1       0
1      NaN        NaN            NaN            NaN       100   6% Bond

Index Type    UnderInd OptSpec Strike ExerciseDates  AmericanOpt Name
4     OptBond 3        Call    101    01-Jan-2003    0           Option
```

下面的代码将证券数据储存在 Excel 工作表中的变量并且创建一个工具集进行处理.

```
Sub PutInstData()
' Send portfolio data and build InstSet

    'Send Bond data first
    mlPutMatrix "BondName", Worksheets("HJMDemo").Range("BondName")
    mlPutMatrix "BondRate", Worksheets("HJMDemo").Range("BondRate")
    mlPutMatrix "BondSettle", Worksheets("HJMDemo").Range("BondSettle")
    mlPutMatrix "BondMaturity", Worksheets("HJMDemo").Range("BondMaturity")
    mlPutMatrix "BondPeriod", Worksheets("HJMDemo").Range("BondPeriod")

    'Send Option
    mlPutMatrix "OptType", Worksheets("HJMDemo").Range("OptType")
    mlPutMatrix "OptUndBond", Worksheets("HJMDemo").Range("OptUndBond")
    mlPutMatrix "OptionStrike", Worksheets("HJMDemo").Range("OptionStrike")
    mlPutMatrix "OptionExDates",
      Worksheets("HJMDemo").Range("OptionExDates")
    mlPutMatrix "OptionAmEu", Worksheets("HJMDemo").Range("OptionAmEu")

    'Excel Serial Date Number Form to MATLAB Serial Date Number Form
    mlevalstring "BondSettle = x2mdate(BondSettle);"
    mlevalstring "BondMaturity = x2mdate(BondMaturity);"
    mlevalstring "OptionExDates = x2mdate(OptionExDates);"

    'Create InstSet
    mlevalstring "ProcessInst"

End Sub

Sub PutFwdCurve()
 'Send the initial forward curve to MATLAB and calculate RateSpec.

    mlPutMatrix "FwdStartDates",
      Worksheets("HJMDemo").Range("FwdStartDates")
    mlPutMatrix "FwdEndDates", Worksheets("HJMDemo").Range(" FwdEndDates")
    mlPutMatrix "FwdRates", Worksheets("HJMDemo").Range("FwdRates")

    'Excel Serial Date Number Form to MATLAB Serial Date Number Form
    mlevalstring "FwdStartDates = x2mdate(FwdStartDates);"
    mlevalstring "FwdEndDates = x2mdate(FwdEndDates);"

    'mlevalstring "ProcessFwdCurve"
    mlevalstring "RateSpec = intenvset('Rates', FwdRates, 'StartDates',
      FwdStartDates, 'EndDates', FwdEndDates,'Compounding', 1);"

End Sub
```

```
Sub PutSigma()

 'Send sigma to MATLAB.
 mlPutMatrix "Sigma", Worksheets("HJMDemo").Range("Sigma")

End Sub

Sub CalculatePrices()

    Call CalculateResults(False)

    Call MLGetMatrix("PriceBond", "G14")
    Call MLGetMatrix("PriceOpt", "G21")
    MatlabRequest

End Sub

Sub ViewPriceTree()

'View the price tree
    Call CalculateResults(True)

End Sub

Sub CalculateResults(bPlot As Boolean)

    'Create TimeSpec
    mlevalstring "TimeSpec = hjmtimespec(RateSpec.ValuationDate,
      RateSpec.EndDates(1:end), RateSpec.Compounding);"
    'Create VolSpec
    mlevalstring "VolSpec = hjmvolspec('Constant', Sigma);"
    'Build tree
    mlevalstring "HJMTree = hjmtree(VolSpec,RateSpec,TimeSpec);"


    If (bPlot) Then
        'Compute price/view tree
        mlevalstring "[Price, PriceTree] = hjmprice(HJMTree, InstSet);"
        mlevalstring "treeviewer(PriceTree,InstSet);"
    Else
         'Compute price
         mlevalstring "[Price] = hjmprice(HJMTree, InstSet);"
         mlevalstring "PriceBond =Price(1:3);"
         mlevalstring "PriceOpt =Price(end);"
    End If

End Sub
```

要在 Matlab 中处理这些数据，调用下列代码：

```
%-------------------------
% Process the bond data:
InstSet = instadd('Bond', BondRate(1), BondSettle(1), BondMaturity(1),
            BondPeriod(1));
for i=2:length(BondName)
        InstSet = instadd(InstSet, 'Bond', BondRate(i), BondSettle(i),
                    BondMaturity(i), BondPeriod(i));
end

% Process option
% First find option index
```

```
BondIndex = strmatch(OptUndBond, BondName);
% If found, add to portfolio
if ~isempty(BondIndex)
      InstSet = instadd(InstSet, 'OptBond', BondIndex, OptType,
                  OptionStrike, OptionExDates, OptionAmEu);
end

InstSet = instsetfield(InstSet, 'Index',1:4, 'FieldName', {'Name'}, ...
                  'Data',  {'4% Bond'; '5% Bond'; '6% Bond';'Option'
});

clear i
```

A.6 双因素 HJM 模型用 Matlab 实现

下面是双因素 HJM 树构建远期利率的一个应用. 假设两个因素都有由 Vasicek 指数波动率函数 $\sigma_i e^{-\lambda_i(T-t)}$ ($i=1$, 2)产生的波动率. 首先用 hjmvolspec 指定这个波动率函数.

```
VolSpec = hjmvolspec('Exponential', 0.1, 1, 'Exponential', 0.5,1)

FinObj: 'HJMVolSpec'
FactorModels: {'Exponential'  'Exponential'}
FactorArgs: {{1x2 cell}  {1x2 cell}}
SigmaShift: 0
NumFactors: 2
NumBranch: 3
PBranch: [0.2500 0.2500 0.5000]
Fact2Branch: [2x3 double]
```

然后为固定收益衍生品定价:

```
VolSpec = hjmvolspec('Exponential', 0.1, 1, 'Exponential', 0.5,1)
RateSpec = intenvset('Compounding', Compounding,...
                  'ValuationDate', ValuationDate,...
                  'StartDates', StartDate,...
                  'EndDates', EndDates,...
                  'Rates', Rates);
TimeSpec = hjmtimespec(ValuationDate, EndDates, Compounding);
HJMTree = hjmtree(VolSpec, RateSpec, TimeSpec);
[Price, PriceTree] = hjmprice(HJMTree, RateSpec, TimeSpec)
```

双因素 HJM 树在 C++ 中的实现, 参见 London(2004).

尾注

1. Jamshidian F(1991), "Bond and Option Evaluation in the GaussianInterest Rate Model." *Research in Finance* 9: 131 - 170.
2. Hull J, White A(1994), "Numerical Procedures for Implementing Term Structure Models I: Single Factor Models," *Journl of Derivatives*, 1: 7 - 16.
3. 在许多书籍和论文中, 模型写为更常用的形式: $dr=(\theta(t)-r)dt+\sigma\, dz$, 其中 $\theta(t)=a\bar{r}(t)$.
4. Δr 与 Δt 间的关系被认为是好的选择, 因为其满足三项式法的稳定收敛条件, 与显式有限差分结构一致.
5. Black F, Karasinski P (1991), "Bond and Option Pricing When Short Rates Are Lognormal." *Financial Analysts Jourmal*, July: 52 - 59.
6. Heath D, Jarrow R, Morton A(1992), "Bond Pricing and the Term Structure of Interest Rates: A New Methodology for Contingent Claims Vsluation." *Econometrica*, 60(1): 77 - 105.

附录 B　第 7 章的代码

鉴于第 7 章代码文件的长度，本书把 Matlab 代码附在这个附录里. 本附录包含了 7.7 节中 Doerr(2003) 为摆动期权定价的代码，7.13 节中 Xiang(2004) 电力扩散模型的极大似然参数估计，以及 7.17 节中 Xu(2004) 天然气期权的定价.

摘自 7.7 节，Matlab 中摆动期权定价

下面的 Matlab 代码，由 Doerr(2004) 撰写，提供了 LSM 算法使用向上摆动、向下摆动和惩罚函数为摆动期权定价的应用.

Find_strategy.m

```
function f = find_strategy(cf_matrix,exercises,timesteps,strikes)
%calculates strategy matrix from given cashflow matrix
strat = zeros(exercises,timesteps);
strat(:,timesteps) = strikes(timesteps)*ones(exercises,1);
for e=2:exercises
   strat(:,timesteps+1-e) = cat(1,strikes(timesteps+1-e)*
     ones(exercises+1-e,1),zeros(e-1,1));
end
num_exercise(:,1) = (cf_matrix(:,1)~=0);
for i=2:timesteps
   num_exercise(:,i) = num_exercise(:,i-1) + (cf_matrix(:,i)~=0) ;
end
for i=1:timesteps
   num = num_exercise(:,i);
   cf = cf_matrix(:,i);
   num_exercise(:,i) = num.*(cf~=0);
end
%c = cf_matrix;
%n = num_exercise;
for i=1:exercises
   for j=i:timesteps-1-exercises+i
      cf_vector = cf_matrix(:,j);
      num_vector = num_exercise(:,j);
      a = (num_vector==i);
      if norm(a)==0
         strat(i,j) = 0;
      else
         strat(i,j) = min(cf_vector(num_vector==i)) + strikes(j);
      end
   end
end
cf_matrix;
f = strat;
```

Coinc_strat_upswing.m

```
function swi = coinc_strat_upswing(paths,timesteps,delta_t,start_value,f,
  alpha,sigma,exercises,strikes)

loops = 10;
av = 0;
sqr_av = 0;
for l=1:loops
```

```
        S = create_paths_anti(paths,timesteps,delta_t,start_value,f,
          alpha,sigma);
        all_comb = combnk(1:timesteps,exercises);
        [c dummy] = size(all_comb);
        ex = unidrnd(c,paths,1);
        ex_ind = all_comb(ex);
        for i=2:exercises
        col_vector = all_comb(:,i);
        ex_ind = cat(2,ex_ind,col_vector(ex));
        end
        for i=1:timesteps
        ex_matrix(:,i) = zeros(paths,1);
        for e=1:exercises
        ex_matrix(:,i)=ex_matrix(:,i)+(ex_ind(:,e)==i);
        end
        end
        strike = repmat(strikes,paths,1);
        payoff = max(S-strike,0);
   cf = payoff.*ex_matrix;
   sim_value = sum(sum(cf'))/paths;
   av = av + sim_value;
   sqr_av = sqr_av + sim_value*sim_value;
end
av = av/loops;
sqr_av = sqr_av/loops;
rmse = sqrt(sqr_av-av*av);
swi = [av rmse];
```

Cont_values.m

```
function c = cont_values(payoff,prices,cfws)

[zeile spalte wert] = find(payoff);
[m n] = size(payoff);
red_prices = prices(payoff>0);
red_cf = cfws(payoff>0);
p = polyfit(red prices,red_cf,2);
red_cont = polyval(p,red_prices);
cont = sparse(zeile,spalte,red_cont,m,1);
c = full(cont);
```

Creat_paths_anti.m

```
function cp=create_paths_anti(paths,timesteps,delta_t,start_value,f,
  alpha,sigma);
%creates paths for electricity spot prices following a one-factor
%mean-reverting process
halfpaths = floor(paths/2);
rand_num = randn([halfpaths timesteps]);
anti = -rand_num;
full_randnum = cat(1,rand_num,anti);
%var = 1/(2*alpha)*(exp(2*alpha*delta_t)-1);
vari = sigma*sigma/(2*alpha)*(1-exp(-2*alpha*delta_t));
X = full_randnum*diag(sqrt(vari));
%Y = exp(sigma*X);
Y = exp(X);
beta = ones(1,timesteps)*f^diag(1-exp(-alpha*delta_t));
A = Y*diag(beta);
P = start_value*ones(paths,1);
S = start_value*ones(paths,timesteps);
for i=1:timesteps
   P = (P.^(exp(-alpha*delta_t(i)))).*A(:,i);
```

```
    S(:,i) = P;
end
L = log(S);
%cp=[mean(L) var(L)];
cp = S;
```

Strat_upswing.m

```
function swi = strat_upswing(paths,timesteps,delta_t,start_value,f,alpha,
  sigma,exercises,strikes,strat_matrix,loops)
%calculates value of swing option by applying a given strategy matrix
av = 0;
sqr_av = 0;
for l=1:loops
        S = create_paths_anti(paths,timesteps,delta_t,start_value,f,
          alpha,sigma);
        cf_matrix = zeros(paths,timesteps);
        ex_ind = zeros(paths,1);
        for i=1:timesteps
        pcs = S(:,i);
        str = strikes(i)*ones(paths,1);
        pyf = pcs - str;
        for j=1:min(i,exercises)
              r = (pcs>strat_matrix(j,i)) & (strat_matrix(j,i)~=0) &
              (ex_ind==j-1);
              cf_matrix(:,i) = cf_matrix(:,i) + r.*pyf;
        end
        ex_ind = ex_ind + (cf_matrix(:,i)>0);
        end
   val_vector = sum(cf_matrix');
   av_value = sum(val_vector)/paths;
   av = av + av_value;
   sqr_av = sqr_av + av_value*av_value;
end
av = av/loops;
sqr_av = sqr_av/loops;
rmse = sqrt(sqr_av-av*av);
swi = [av rmse];
```

ls_upswing_strat.m

```
function ls = ls_upswing_strat(paths,timesteps,delta_t,start_value,f,
  alpha,sigma,exercises,strikes)
%calculates value of a swing option using extended L+S; strategy matrix
%is calculated in addition only upswings, no penalty!!
loops = 5;
av=zeros(exercises,1);
sqr_av=zeros(exercises,1);
av_strat = zeros(exercises,timesteps);
sqr_strat = zeros(exercises,timesteps);
for s=1:loops
   S = create_paths_anti(paths,timesteps,delta_t,start_value,f,alpha,sigma);
        cashflow = zeros(paths,timesteps,exercises);
        cont = zeros(paths,exercises);
        for i = 1:timesteps
        strike = strikes(timesteps+1-i)*ones(paths,1);
        pcs = S(:,timesteps+1-i);
        pyf = max(pcs,strike)-strike;
        for e = 1:exercises
                if e >= i
                cashflow(:,timesteps+1-i,e) = pyf;
        else
```

```
                if i > 2
                cf = sum(cashflow(:,timesteps+2-i:timesteps,e)')';
                else
                cf = cashflow(:,timesteps,e);
                end
                cont(:,e) = cont_values(pyf,pcs,cf);
        end
        end
        if i>1
                cashflow = reset_cfmatrix(cashflow,pyf,cont,i,timesteps,
                  exercises);
        end
    end
    for e=1:exercises
         valvector = sum(cashflow(:,:,e)')';
         av_value(e) = sum(valvector)/paths;
         av(e) = av(e)+av_value(e);
       sqr_av(e) = sqr_av(e)+av_value(e)*av_value(e);
    end
    strat_matrix = find_strategy(cashflow(:,:,exercises),exercises,
      timesteps,strikes);
    av_strat = av_strat + strat_matrix;
    sqr_strat = sqr_strat + strat_matrix.*strat_matrix;
end
av=av/loops;
sqr_av=sqr_av/loops;
rmse=sqrt(sqr_av-av.*av);
strat = av_strat/loops;
sqr_strat = sqr_strat/loops;
rmse_strat = sqrt(sqr_strat-strat.*strat);
%ls = strat;
ls=[av rmse strat rmse_strat];
```

Is_upswing.m

```
function ls = ls_upswing(paths,timesteps,delta_t,start_value,f,alpha,sigma,
  exercises,strikes)

loops = 5;
av=zeros(exercises,1);
sqr_av=zeros(exercises,1);
for s=1:loops
   S = create_paths_anti(paths,timesteps,delta_t,start_value,f,alpha,sigma);
   cashflow = zeros(paths,timesteps,exercises);
        cont = zeros(paths,exercises);
        for i = 1:timesteps
        strike = strikes(timesteps+1-i)*ones(paths,1);
        pcs = S(:,timesteps+1-i);
        pyf = max(pcs,strike)-strike;
        for e = 1:exercises
                if e >= i
                cashflow(:,timesteps+1-i,e) = pyf;
        else
                if i > 2
                cf = sum(cashflow(:,timesteps+2-i:timesteps,e)')';
                else
                cf = cashflow(:,timesteps,e);
                end
                cont(:,e) = cont_values(pyf,pcs,cf);
        end
```

```
        end
        if i>1
                cashflow = reset_cfmatrix(cashflow,pyf,cont,i,timesteps,
                  exercises);
        end
    end
    for e=1:exercises
        valvector = sum(cashflow(:,:,e)')';
        av_value(e) = sum(valvector)/paths;
        av(e) = av(e)+av_value(e);
      sqr_av(e) = sqr_av(e)+av_value(e)*av_value(e);
    end
end
av=av/loops;
sqr_av=sqr_av/loops;
rmse=sqrt(sqr_av-av.*av);
ls=[av rmse];
```

ls_updownswing.m

```
function ls = ls_updownswing(paths,timesteps,delta_t,start_value,f,alpha,
  sigma,ups,ustrikes,downs,dstrikes)

loops = 5;
av = 0;
sqr_av = 0;
for s=1:loops
   S = create_paths_anti(paths,timesteps,delta_t,start_value,f,alpha,sigma);
   % cashflow(:,:,u,d) means u-1 upswings and d-1 downswings are
   % already exercised!!
   cashflow = zeros(paths,timesteps,ups+1,downs+1);
   cont = zeros(paths,1,ups+1,downs+1);
   % initial timestep
   pcs = S(:,timesteps);
   t = timesteps;
   upyf = max(pcs-ustrikes(t),0);
   dpyf = max(dstrikes(t)-pcs,0);
   % both upswings and downswings left
   for u=0:ups-1
      for d=0:downs
         cashflow(:,t,u+1,d+1) = max(max(upyf-penalty(u-d+1),
           dpyf-penalty(u-d-1)),-penalty(u-d));
      end
   end
   % no downswings, but still upswings left
   for u=0:ups-1
      cashflow(:,t,u+1,downs+1) = max(upyf-penalty(u+1-downs),
        -penalty(u-downs));
   end
   % no upswings, but still downswings left
   for d=0:downs-1
      cashflow(:,t,ups+1,d+1) = max(dpyf-penalty(ups-d-1),-penalty(ups-d));
   end
   % no upswings and no downswings left
   cashflow(:,t,ups+1,downs+1) = -penalty(ups-downs);
   % stepping backwards in time
   for i=2:timesteps
        i;
        pcs = S(:,timesteps+1-i);
      upyf = max(pcs-ustrikes(timesteps+1-i),0);
      dpyf = max(dstrikes(timesteps+1-i)-pcs,0);
```

```
        % calculate continuation values
        ustop = min(ups,timesteps+1-i);
        for u=0:ustop
           for d=0:min(downs,timesteps+1-i-u)
              u;
              d;
                  if i > 2
                          cf = sum(cashflow(:,timesteps+2-i:timesteps,
                            u+1,d+1)')';
                  else
                          cf = cashflow(:,timesteps,u+1,d+1);
              end
              cont(:,1,u+1,d+1) = cont_values(upyf,pcs,cf) +
                cont_values(dpyf,pcs,cf);
           end
        end
        cashflow = reset_cfmatrix_ud(cashflow,pcs,ustrikes(i),dstrikes(i),
          cont,i,timesteps,ups,downs);
    end
    cashflow(:,:,1,1);
    valvector = sum(cashflow(:,:,1,1)')';
          av_value = sum(valvector)/paths;
    av = av + av_value;
    sqr_av = sqr_av + av_value*av_value;
end
av=av/loops;
sqr_av=sqr_av/loops;
rmse=sqrt(sqr_av-av.*av);
ls=[av rmse];
```

reset_cfmatrix.m

```
function ncf=reset_cfmatrix(old_cfmatrix,payoff,contmatrix,timestep,
  timesteps,exercises)
% timestep is counted backwards!!
[m dummy] = size(payoff);
for e=timestep:exercises
   new_cfmatrix(:,:,e) = old_cfmatrix(:,:,e);
end
stop = min(timestep-1,exercises);
for c=1:stop
   e = stop+1-c;
   if e==1
      z = find(payoff>contmatrix(:,1));
      wert = payoff(z);
   else
      z = find(payoff+contmatrix(:,e-1)>contmatrix(:,e));
      wert = payoff(z);
   end
   for j=timesteps-timestep+2:timesteps
      old_cfvector = old_cfmatrix(:,j,e);
      red_oldcf = full(sparse(z,1,old_cfvector(z),m,1));
      if e==1
         new_cfvector = zeros(m,1);
      else
         new_cfvector = old_cfmatrix(:,j,e-1);
      end
      red_newcf = full(sparse(z,1,new_cfvector(z),m,1));
      new_cfmatrix(:,j,e) = old_cfmatrix(:,j,e)-red_oldcf+red_newcf;
   end
```

```
    new_cfmatrix(:,timesteps+1-timestep,e) = full(sparse(z,1,wert,m,1));
  end
  ncf = new_cfmatrix;
```

reset_cfmatrix_ud.m

```
function new = reset_cfmatrix_ud(old_cashflow,pcs,ustrike,dstrike,
  contmatrix,i,timesteps,ups,downs)
%i is timestep counted from back
[paths dummy] = size(pcs);
new_cashflow = old_cashflow;
upyf = max(pcs-ustrike,0);
dpyf = max(dstrike-pcs,0);
ustop = min(ups-1,timesteps-i);
%both upswings and downswings left
for u=0:ustop
   for d=0:min(downs-1,timesteps-i-u)
      %exercise upswing
      cont_noex = contmatrix(:,1,u+1,d+1);
      cont_ex = contmatrix(:,1,u+2,d+1);
      old_cf = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
        u+1,d+1));
      cf_ex = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
        u+2,d+1));
      ncf_up = exercise(upyf,cont_noex,cont_ex,old_cf,cf_ex);
      %exercise downswing
      cont_noex = contmatrix(:,1,u+1,d+1);
      cont_ex = contmatrix(:,1,u+1,d+2);
      old_cf = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
        u+1,d+1));
      cf_ex = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
        u+1,d+2));
      ncf_down = exercise(dpyf,cont_noex,cont_ex,old_cf,cf_ex);
      %put together up and down
      new_cashflow(:,timesteps+1-i:timesteps,u+1,d+1) =
        merge_updown(ncf_up,ncf_down,old_cf);
   end
end
%no upswings left
for d=0:min(downs-1,timesteps-i-ups)
   cont_noex = contmatrix(:,1,ups+1,d+1);
   cont_ex = contmatrix(:,1,ups+1,d+2);
   old_cf = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
     ups+1,d+1));
   cf_ex = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
     ups+1,d+2));
   new_cashflow(:,timesteps+1-i:timesteps,ups+1,d+1) = exercise(dpyf,
     cont_noex,cont_ex,old_cf,cf_ex);
end
%no downswings left
for u=0:min(ups-1,timesteps-i-downs)
   cont_noex = contmatrix(:,1,u+1,downs+1);
   cont_ex = contmatrix(:,1,u+2,downs+1);
   old_cf = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
     u+1,downs+1));
   cf_ex = cat(2,zeros(paths,1),old_cashflow(:,timesteps+2-i:timesteps,
     u+2,downs+1));
   new_cashflow(:,timesteps+1-i:timesteps,u+1,downs+1) = exercise(upyf,
     cont_noex,cont_ex,old_cf,cf_ex);
end
new = new_cashflow;
```

penalty.m

```
function p=penalty(v)
if v~=0
   p = 1000000000;
else
        p = 0;
end;
```

主要的功能驱动程序如下：

staple_paper.m

```
function st = stapel_paper(paths)
alpha = 0.55;
sigma = 0.95;
timesteps = 7;
delta_t = [1 1 1 1 1 1 1];
f = 22.5406;
strikes = f*delta_t;
start_values = [0.1 1 5 10 15 20 25 30 40 60 100];

res=zeros(1,3);
for sta=1:11
        ind = start_values(sta);
   act_res = ls_updownswing(paths,timesteps,delta_t,start_values(sta),f,
     alpha,sigma,3,strikes,3,strikes);
   act_erg = cat(2,ind,act_res);
   res = cat(1,res,act_erg);
end
for sta=1:11
        ind = start_values(sta);
   act_res = ls_upswing(paths,timesteps,delta_t,start_values(sta),f,alpha,
     sigma,3,strikes);
   act_erg = cat(2,ind,act_res(3,:));
   res = cat(1,res,act_erg);
end
for sta=1:11
        ind = start_values(sta);
   act_res = coinc_strat_upswing(paths,timesteps,delta_t,start_values(sta),
     f,alpha,sigma,3,strikes);
   act_erg = cat(2,ind,act_res);
   res = cat(1,res,act_erg);
end
st = res;
```

staple.m

```
function st = stapel(paths,timesteps,delta_t,f,ups,downs,strikes)

%[za sa] = size(alphas);
%[zs ss] = size(start_values);
%res = zeros(exercises,1);
%for a = 1:za
%   for s = 1:zs
%      res = cat(2,res,ls_upswing(paths,timesteps,delta_t,start_values(s),
%        f,alphas(a),sigma,exercises,strikes));
%   end
%end
%st = res;
alphas = [0.05,0.5];
```

```
start_values = [0.01,1,5,10,15,20,30,40,50,60,70,80];
sigmas = [0.392];

res = zeros(1,3);
for a=1:2
   for sig=1:1
      for sta=9:12
         ind(1,1) = start_values(sta);
                ind(1,2) = alphas(a);
         act_res = ls_updownswing(paths,timesteps,delta_t,start_values(sta),
           f,alphas(a),sigmas(sig),ups,strikes,downs,strikes);
         zahl = act_res(1,1);
         act_erg = cat(2,ind,zahl);
                res = cat(1,res,act_erg);
      end
   end
end

for a=1:2
   for sig=1:1
      for sta=9:12
         ind(1,1) = start_values(sta);
            ind(1,2) = alphas(a);
         act_res = ls_updownswing_p2(paths,timesteps,delta_t,
           start_values(sta),f,alphas(a),sigmas(sig),ups,strikes,
           downs,strikes);
         zahl = act_res(1,1);
         act_erg = cat(2,ind,zahl);
                res = cat(1,res,act_erg);
      end
        end
end

st = res;
```

摘自 7.13 节，Matlab 中参数估计

下面的 Matlab 代码由 Lei Xiong 和 Anthony Ware 编写，用前一节中给出的 Xiong(2004) 的极大似然方法估计模型参数. Xiong 和 Ware 用来校准和估计模型参数的市场数据（储存在数组里在函数中使用）来自于 Alberta Nordic Pool（这里没有提供）.

在进行估计之前，数据需要做季节性调节.

fixdata.m

```
% This is a utility function to help prepare the
% data, for example by detrending, or in the case of hourly electricity
% data, removing average daily fluctuations. It also offers the choice
% to randomize the data in order to compensate for quantisation effects
% should the user wish to do so
% datafile: the name of the data
% makerandom: randomization factor (0 means no randomization).
% deseasonalize: 1 if you want to get rid of seasonal factors
% y: rearranged logarithm of spot price
% Dt: annualized time step
%    e.g. 1/365 for daily gas data, 1/252 for daily oil data
% new_mdata: it is empty if no forward price is supplied.
%   Otherwise, it is arranged in 5 columns. The first is the
%   market date (t), the second is the delivery start date(T),
```

```
%   the third is the delivery end date (T+M), the fourth is
%   the forward price, and the fifth is the spot price for
%   that day.

function [y,Dt] = fixdata(datafile,makerandom,deseasonalize)

new_mdata = [];

switch datafile
 case 'pooltest'
        load pooltest
        N = length(p);
        pl = p(:,1); pl = pl(:);
        %pl = randomize(pl,makerandom);
        Ndays = floor(N/24);
         N = Ndays*24;
        pl = reshape(pl(1:N),24,Ndays);
        pl = log(pl);
        s = [];
        if deseasonalize
        s = mean(pl,2);
            y = pl - repmat(s,1,Ndays);
        else
            y=pl;
        end
        Dt = 1/(24*365);
        pl = y(:);
        p(:,1) = pl;
        y = p;
 case 'onpeak'
        load onpeak
        p = log(onpeak);
        y = p(:);
        Dt = 1/365;
 case 'offpeak'
        load offpeak
        p = log(offpeak);
        y = p(:);
        Dt = 1/365;
 case 'paverage'
        load paverage
        p = avep;
        p = log(p);
        y = p(:);
        Dt = 1/365;
 end
```

inittheta.m

```
% This function gets the initial transformed theta (see "transtheta")
% together with the initial theta.
% y : the data you deal with
% Nseason:
%    -1: with no trend and no seasonal factor
%           0: with trend but no seasonal factor
%     1: with trend and one seasonal factor
%        (with a seasonal cycle as a year)
%     2: with trend and two seasonal factors
%        (with a seasonal cycle as half a year)
%     3: with trend and three seasonal factors
%        (with a seasonal cycle as a quarter)
```

```
% getdist:
%     1: the jump component has exponentially distributed
%        absolute value of jump size with a sign of the jump
%        determined by a Bernoulli variable.
%     2: the jump component is log-normal.
% Ttheta0: transformed initial theta
% theta0: initial theta

function [Ttheta0,theta0] = inittheta(y,getdummy,getdist)

if nargin < 2,
  Nseason = -1;
end

if nargin < 3,
  getdist = 1;
end

B = y(:,2)-y(:,1);
C = [-y(:,1) ones(size(y(:,1)))];
x = lsqlin(C,B);
thetapart = zeros(1,6);
thetapart(2) = x(2)/x(1);
thetapart(1) = x(1);
thetapart(3) = norm(C*x-B)^2/(size(y,1));
datadiff = y(:,2)-y(:,1);
switch getdist
  case 1
    thetapart(4) = 0.45;
    thetapart(5) = 0.45;
    thetapart(6) = mean(abs(datadiff));
  case 2
    upind = find(datadiff>0);
    downind = find(datadiff<0);
    thetapart(4) = 0.5;
    thetapart(5) = mean(datadiff(upind));
    thetapart(6) = 0.5;
thetapart = [thetapart -mean(datadiff(downind))];
end

switch getdummy
  case 'n'
     theta0 = thetapart;
  case 'y'
     theta0 = [thetapart,thetapart(2)];
end

Ttheta0 = transtheta(theta0,1,getdist);
```

GeeV. m

```
function [gv,N,R,fftT,Tgrad] = GeeV(Ttheta,fftT,getdist)
% This function helps to compute the integral in the density function by
% fast Fourier transform algorithm, called by function "LogExponential"
% Ttheta: the transformed theta
% fftT: is the interval of FFT
% gv: result obtained from the FFT computation
% N: the number of points of FFT computation
% R: the range of the the integration
% Tgrad: the gradient matrix w.r.t `Ttheta'

dt = 1;
```

```
dograd = (nargout>4);
TOL = 1e-5;
Nmax = 2^13;
% the allowed maximum value of N
theta = transtheta(Ttheta,-1,getdist,dt);
k = theta(1);
% k is kappa, the coeffecient of mean reversion
a = theta(2);
% a is alpha, long term mean of logged the price
sigma2 = theta(3);
% sigma^2 volitility, var(dp_t)
w = theta(4);
% omega, poisson arrival rate
psi = theta(5); % sign of the jump, it's a Bernoulli's parameter 0<psi<1
r = theta(6);
% r is gamma, the magnitude of jump.
if  any(imag(Ttheta)) | any(isinf(Ttheta))|k==0
  gv = []; N = [];  R = []; fftT = [];  Tgrad = [];
  return
end

% -----------------find a suitable value for R---------------------------
R2 = 10;
expR2 = (-(R2*sigma2/(4*k))*(1-exp(-2*k)) + (w/(2*k))* ...
    log((1+(r^2)*R2*exp(-2*k))./(1+(r^2)*R2)));
count = 0;
while expR2 > log(TOL);
count = count+1;
R2 = 2*R2 ;
expR2 = (-(R2*sigma2/(4*k))*(1-exp(-2*k)) + (w/(2*k))* ...
        log((1+(r^2)*R2*exp(-2*k))./(1+(r^2)*R2)));
end

R = sqrt(R2);
%------------------find a suitable value for fftT and N ----------------
N = 2 ^ ceil( 1+log2( R*fftT/pi ) );
fftT = N*pi/(2*R);
isNtoobig = 0;
if log2(N)> 18
  %disp(['N too large; R2=',num2str(R2),'; k=',num2str(k),'; s2=',
  %num2str(sigma2)]);
isNtoobig = 1;
gv = []; N = [];  R = []; fftT = [];  Tgrad = [];
    return
end
doagain = 1;
while doagain==1
    ds = 2*R/N;
    ss = linspace(-R,R-ds,N);
    ss = ss([(N/2)+1:N,1:N/2]);
    A1 = -((ss.^2)*sigma2/(4*k))*(1-exp(-2*k));
    A2 = (1i*w*(1-2*psi)/(k))*(atan(ss*exp(-k)*r)-atan(ss*r));
    A3 = (w/(2*k))*log((1+(ss.^2)*exp(-2*k)*(r^2))./(1+(ss.^2)*(r^2)));
    h = exp(A1+A2+A3);    % see formula 26 in the document
    g1 = (1/(2*pi))*ds*fft(h);
    g=[g1((N/2+1):end),g1(1:N/2+1)];
    gv = real(g);

  if max(abs(gv(1)))<TOL*max(gv)
    doagain=0;
```

```
      else

       N = 2*N; fftT = 2*fftT;


       if log2(N)>18
         %disp(['N too large; R2=',num2str(R2),'; k=',num2str(k),'; s2=',
         %num2str(sigma2)]);
         isNtoobig = 1;
         gv = []; N = [];  R = []; fftT = [];  Tgrad = [];
         return
       end
     end
   end

   [tmp,i]=max(gv);
   if i==1

   gv = []; N = [];  R = []; fftT = [];  Tgrad = [];
   return
end
errest = abs(mean(gv([i-1,i+1])-gv(i)));

while errest > TOL*gv(i) & N<Nmax
    N = 2*N;
    R = 2*R;
    ds = 2*R/N;
    ss = linspace(-R,R-ds,N);
    ss = ss([(N/2)+1:end,1:N/2]);
    A1 = -((ss.^2)*sigma2/(4*k))*(1-exp(-2*k));
    A2 = (1i*w*(1-2*psi)/(k))*(atan(ss*exp(-k)*r)-atan(ss*r));
    A3 = (w/(2*k))*log((1+(ss.^2)*exp(-2*k)*(r^2))./(1+(ss.^2)*(r^2)));
    h = exp(A1+A2+A3);
    % h = h(s).
    g1 = (1/(2*pi))*ds*fft(h);
    g=[g1((N/2+1):end),g1(1:N/2+1)];
    gv = real(g);
    [tmp,i]=max(gv);
    errest = abs(mean(gv([i-1,i+1])-gv(i)));
end

%------------------Evaluate the gradient-----------------------------------
if dograd
  % to compute gradient we'll take the fft of  s.*h, (s.^2).*h,
  % (atan(...)...).*h
  % ds/pi = R/(N*pi) = 1/fftT and this is the mean of z!
  % f_{j}(theta) = g(t,theta) = 1/2\pi \int^{\infty}_{\infty} Q(s,\theta)...
  % exp(-1i*s*t) ds and t= (y_{j} -a) -(y_{j-1}-a)*exp(-k)
  % evaluate gradient

  if k~=0|w~=0
    A1k = -A1/k - (ss.^2)*sigma2*exp(-2*k)/(2*k);
    A2k = -A2/k - 1i*w*(1-2*psi)*r*exp(-k)*ss./(k*(1+r^2*ss.^2*exp(-2*k)));
    A3k = -A3/k -w*r^2*ss.^2*exp(-2*k)./(k*(1+r^2*ss.^2*exp(-2*k)));
    fk2 = (1/(2*pi))*fft((A1k+A2k+A3k).*h)*ds;
    fs2 = (1/(2*pi))*fft(ss.^2.*h)*ds;
    A2w = A2/w;
    A3w = A3/w;
    fw = (1/(2*pi))*fft((A2w+A3w).*h)*ds;
```

```
    %derivative respect to psi
    A2psi = -(2*1i*w/k)*(atan(ss*exp(-k)*r)-atan(ss*r));
    fpsi = (1/(2*pi))*fft(A2psi.*h)*ds;
    fs1 = -1i*(1/(2*pi))*fft(ss.*h)*ds;

    %derivative respect to gamma
    A2r = 1i*w*(1-2*psi)/k*(exp(-k)*ss./(1+r^2*ss.^2*exp(-2*k))-ss./
      (1+r^2*ss.^2));
    A3r = w/k*(r*exp(-2*k)*ss.^2./(1+r^2*ss.^2*exp(-2*k))-r*ss.^2./
      (1+r^2*ss.^2));
    fr = (1/(2*pi))*fft((A2r+A3r).*h)*ds;
    gradtmp1 = [fs1; fk2; fs2; fw; fpsi; fr];
    gradtmp2 = [gradtmp1(:,(N/2+1):end), gradtmp1(:,(1:N/2+1))];
    Tgrad =real(gradtmp2);
  else
   Tgrad =[];
   return
  end
else
  Tgrad =[];
```

LogExponential. m

```
function [MinLLhood,Tgradient,gv,N,R,fftT] = LogExponential(Ttheta,p,
  getdist,getdummy,pdummy)
%Get the negative likelihood value for the Log-Exponential Jump

% Ttheta: the transformed theta
% y: rearranged data y_t, y(:,1) = y_t, y(:,2) = y_t+1
% getdist: represents the distribution type, shall be `1'
% Dt: the annualized time step
% onoffind: the index of on-peak or off-peak data
% MinLLhood: minus loglikelihood at the solution
% Tgradient: the gradient of the function at `Ttheta'
% gv: result obtained from the FFT computation
% N: the number of points of FFT computation
% R: the range of the the integration
% fftT: the interval of FFT computation

if nargin<5, pdummy = []; end
n = length(p)-1;
dt =1;
dograd = (nargout>1);
NofTh = length(Ttheta);
MinLLhood = 0; Tgradient = [];

theta = transtheta(Ttheta,-1,getdist);
if  any(imag(Ttheta)) | any(isinf(Ttheta))
    MinLLhood = 1e6; gv = []; N = []; R = []; fftT = [];  Tgradient =
      1e-6*ones(NofTh,1);
    return
end

k = theta(1);
a = theta(2);
sigma2 = theta(3);
w = theta(4);
psi = theta(5);
r = theta(6);

%----------- Set up some values for the seasonal factors-----------
```

```
ft = 0;
ft1 = 0 ;% ft is used to adjust the drift by considering the
        % seasonal factors

if strcmp(getdummy,'y') % if the length of theta is larger than 6, we have
                        % included the seasonal factors
        a2 = theta(7);
    if ~isempty(pdummy)
%           ft1 = -a*pdummy(1:end-1,1) - a2*pdummy(1:end-1,2);
%           ft = (-a*pdummy(2:end,1) - a2*pdummy(2:end,2)) - exp(-k)*ft1;
        ft1 = -a*pdummy(2:end,1) - a2*pdummy(2:end,2);
        ft = (-a*pdummy(2:end,1) - a2*pdummy(2:end,2)) - exp(-k)*ft1;
    else
        ft1 = -a*p(1:end-1,2) - a2*p(1:end-1,3);
        ft = (-a*p(2:end,2) - a2*p(2:end,3)) - exp(-k)*ft1;
    end
    t_i_s = (p(2:end,1)) - exp(-k)*(p(1:end-1,1))+ft    ;
else
   t_i_s = (p(2:end,1)-a) - exp(-k)*(p(1:end-1,1)-a)    ;
end
fftT = 2*max(  abs( t_i_s ));
%-----------------------Compute the minus-loglikelihood from FFT---------
 % fftT is the interval of FFT computation
if dograd
    [gv,N,R,fftT,Tgrad] = GeeV(Ttheta,fftT,getdist);
else
    [gv,N,R,fftT] = GeeV(Ttheta,fftT,getdist);
end

if isempty(gv)
    MinLLhood = 1e6; gv = []; N = []; R = []; fftT = []; Tgradient =
      1e-6*ones(NofTh,1);
    return
end

  % the values at which we really want.
t_j_s = linspace(-fftT,fftT,N+1);

if isreal(t_i_s) ~=1 | isnan(gv)
    MinLLhood = 1e6; gv = []; N = []; R = []; fftT = []; Tgradient =
      1e-6*ones(NofTh,1);
    return
else
    fis = interp1(t_j_s,gv,t_i_s,'linear',0);
    if any(fis<=0)
        MinLLhood = 1e6; gv = []; N = []; R = []; fftT = [];  Tgradient =
          1e-6*ones(NofTh,1);
        return
    else
        MinLLhood = -sum(log(fis)); Tgradient = 1e-6*ones(NofTh,1);

        %------------------------------- Evaluate the Tgradient----------.
        if dograd
            temp = zeros(n,6);
            Tgradient = 1e-6*ones(NofTh,1);
            for i = 1:6
                temp(:,i) = interp1(t_j_s,Tgrad(i,:),t_i_s,'linear',0);
            end
```

```
            Tk = exp(-k)*(p(1:end-1,1)-a);
            fT = temp(:,1);
            fk = fT.*Tk + temp(:,2);% Dg/Dk = (P_{j-1}-a)*exp(-k) *
                                   % gt + gk ; k denotes kappa;
            fa = fT.*(exp(-k)-1);
            fsigma2 = 1/(4*k)*(exp(-2*k)-1)*temp(:,3);
            fw = temp(:,4);
            fpsi = temp(:,5);
            fr = temp(:,6);
            gsigma2 = fsigma2*sigma2;
            fk = fk*k;
            gw = fw*w;
            gr = fr*r;
            gpsi = fpsi*0.5*(1-tanh(Ttheta(5))^2);
            gradtemp = [fk,fa ,gsigma2,gw,gpsi,gr];

            %---------------- the following is the Tgradient with respect to
            %---------------- seasonal factors---beta, eta
            if strcmp(getdummy,'y') % if the length of theta is larger
                                    % than 6, we have included the seasonal
                                    % factors
                if ~isempty(pdummy)
%                     fa = fT.*(exp(-k)*pdummy(1:end-1,1) -
%                       pdummy(2:end,1));
%                     fa2 = fT.*(exp(-k)*pdummy(1:end-1,2) -
%                       pdummy(2:end,2));
%                    Tk = exp(-k)*(p(1:end-1,1) - a*pdummy(1:end-1,1) -
%                      a2* pdummy(1:end-1,2));
                    fa = fT.*(exp(-k)*pdummy(2:end,1) - pdummy(2:end,1));
                    fa2 = fT.*(exp(-k)*pdummy(2:end,2) - pdummy(2:end,2));
                   Tk = exp(-k)*(p(1:end-1,1) - a*pdummy(2:end,1) -
                     a2* pdummy(2:end,2));
                else
                    fa = fT.*(exp(-k)*p(1:end-1,2) - p(2:end,2));
                    fa2 = fT.*(exp(-k)*p(1:end-1,3) - p(2:end,3));
                    Tk = exp(-k)*(p(1:end-1,1) - a*p(1:end-1,2) -
                      a2* p(1:end-1,3));
                end
                fk = fT.*Tk + temp(:,2);% Dg/Dk = (P_{j-1}-a)*exp(-k) *
                                       % gt + gk ; k denotes kappa;
                fk = fk*k;
                gradtemp = [fk, fa, gradtemp(:,3:end), fa2];
            end

            for i = 1:NofTh
                Tgradient(i) = -sum(gradtemp(:,i)./fis); % D_{\theta}L =
                                                         % sum (D_{\theta}
                                                         % fi/fi)
            end

        end
    end
end
```

DoEst.m

```
function [Ttheta,MinLLhood,flag,Tgrad,hess] = DoEst(Ttheta0,y,getdist,
  getdummy,pdummy)
% This function is used to get the estimates
% Ttheta0: starting transformed theta
% y: rearranged data y_t, y(:,1) = y_t, y(:,2) = y_t+1
```

```
% getdist: represents the distribution type
% Dt: the annualized time step
% onoffind: the index of on-peak or off-peak data
% Ttheta: the transformed theta
% MinLLhood: minus loglikelihood at the solution
% flag: the exit condition
%    > 0 The function converged to a solution theta;
%    = 0 The maximum number of function evaluations
%        or iterations was exceeded;
%    < 0 The function did not converge to a solution.
% Tgrad: the gradient of the objective function at `Ttheta'
% hess: the value of the Hessian of the objective function
%        at the solution

%Note I changed the tolx from 1e-10 to 1e-6
if nargin<3, getdist = 1;   end

if nargin<5, pdummy =[];   end
Display = 'iter';
  Npoints = 30;

if length(y)>1500
    LargeScale = 'on';
else
    LargeScale = 'off';
end
switch getdist
    case 1
        %         distype = 'LogExponential';
        distype = 'Logdensity';
    case 2
        distype = 'DbExponential';
        %     case 3
        %         distype = 'Logdensity';
end

%-------------------------------First Try-----------------------------------
iternum = 20; % the number of iteration
j = 0;
theta0 = [0.1094   -0.2577    0.0202    0.7669    0.5344    0.2401];
Ttheta0 = transtheta(theta0,1);
opt = optimset('maxfunevals',2000,'tolx',1e-6,'maxiter',iternum, ...
    'gradobj','off','largescale',LargeScale,'display',Display);
[Ttheta, MinLLhood, flag1, output, Tgrad, THess] = fminunc(distype,Ttheta0,
  opt,y,getdist,getdummy,pdummy);
i = 0;
theta = transtheta(Ttheta,-1,getdist);
if flag1 ==1
    switch getdummy
        case 'n'
            if getdist == 1
                transfactor = diag(1./[theta(1),1,theta(3),theta(4),0.5*
                  (1-tanh(Ttheta(5))^2),theta(6)]);
            else
                transfactor = diag(1./[theta(1),1,theta(3),theta(4),
                  theta(5),theta(6),theta(7)]);
            end
        case 'y'
            if getdist == 1
                transfactor = diag(1./[theta(1),1,theta(3),theta(4),0.5*
```

```
                (1-tanh(Ttheta(5))^2),theta(6),1]);
            else
                transfactor = diag(1./[theta(1),1,theta(3),theta(4),
                  theta(5),theta(6),theta(7),1]);
            end
    end
    [TthetaP,maxmllhd,hess] = findpeak(Ttheta,Npoints,THess,distype,y,
      getdist,getdummy,pdummy);
    if maxmllhd <= (-MinLLhood)
        % -MinLLhood is the maximum of likelihood
        flag = flag1;
        return
    end
end
flag = 0;
while  flag<=0 | i<5
    %---------Keep restarting the optimization from last guess----------
    Ntry = 5 ;% Number of restarting times
    while ( flag<=0 ) & j<Ntry
        opt = optimset('GradObj','off', 'maxiter',iternum,'DiffMinChange',
          1e-10,'tolx',1e-6,'largescale',LargeScale,'display',Display,
            'tolfun',1e-7);
        TthetaP = Ttheta*1.005;
        [Ttheta, MinLLhood, flag, output, Tgrad, THess] = fminunc(distype,
          TthetaP,opt,y,getdist,getdummy,pdummy);
        j = j+1;
    end

    if strcmp(LargeScale,'on'), LargeScale = 'off'; else LargeScale =
      'on'; end
    j = 0;
    Ntry = 5; % Number of restarting times
    while ( flag<=0 ) & j<Ntry
        opt = optimset('GradObj','off', 'maxiter',iternum,'DiffMinChange',
          1e-10,'tolx',1e-6,'largescale',LargeScale,'display',Display,
          'tolfun',1e-7);
        TthetaP = Ttheta*1.005;
        [Ttheta, MinLLhood, flag, output, Tgrad, THess] =
          fminunc(distype,TthetaP,opt,y,getdist,getdummy,pdummy);
        j = j+1;
    end

    % -----Keep trying to get the highest points in every slice--------

    [TthetaP,maxmllhd,hess] = findpeak(Ttheta,Npoints,THess,distype,y,
      getdist,getdummy,pdummy); % TthetaP is the Ttheta w.r.t the peak
                                % of those slices

    if maxmllhd <= (-MinLLhood)    % -MinLLhood is the maximum of
                                   % likelihood
        return
    else
        [Ttheta, MinLLhood, flag,output,Tgrad,THess] = fminunc(distype,
          TthetaP,opt,y,getdist,getdummy,pdummy);
    end

    i = i+1;
end
```

findpeak.m

```
function [TthetaP,maxmllhd,hess] = findpeak(Ttheta,Npoints,THess,distype,y,
    getdist,getdummy,pdummy)
%  this function find the peak from slices
%  Npoints is the number of points in one slices
%  TthetaP is the Ttheta w.r.t the peak of those slices

NofTh = length(Ttheta); % the length of theta
llkhds = ones(Npoints,NofTh);
thetaplot = zeros(Npoints,NofTh);
Z = 1.96; % 95 percent confidence
theta = transtheta(Ttheta,-1,getdist);

%~~~~~~~Get the Hessian for untransformed theta~~~~~~~~
switch getdummy
    case 'n'
        if getdist == 1
            transfactor = diag(1./[theta(1),1,theta(3),theta(4),
              0.5*(1-tanh(Ttheta(5))^2),theta(6)]);
        else
            transfactor = diag(1./[theta(1),1,theta(3),theta(4),
              theta(5),theta(6),theta(7)]);
        end
    case 'y'
        if getdist == 1
            transfactor = diag(1./[theta(1),1,theta(3),theta(4),
              0.5*(1-tanh(Ttheta(5))^2),theta(6),1]);
        else
            transfactor = diag(1./[theta(1),1,theta(3),theta(4),
              theta(5),theta(6),theta(7),1]);
        end
end

hess = transfactor * THess * transfactor;
stder = (sqrt(diag(inv(hess))'));
lb = theta - Z*stder;
ub = theta + Z*stder;

for tt = 1:NofTh
    thetatemp = ones(Npoints,1)*theta;
    thetaincrease = linspace(lb(tt),ub(tt),Npoints)';
    thetatemp(:,tt) = thetaincrease;  %thetaincrease records the
                                      %increasement for every parameter
    Tthetatemp = transtheta(thetatemp,1,getdist);
    thetaplot(:,tt) = thetatemp(:,tt);
    for j = 1:Npoints
      llkhds(j,tt) = -feval(distype,Tthetatemp(j,:),y,getdist,
        getdummy,pdummy);
    end
end

% Get the highest points for each slice
ymax = zeros(NofTh,2);
for tt = 1:NofTh
    [ymax(tt,1),ymax(tt,2)] = max(llkhds(:,tt));
end
% Get the maximum of the highest points for all slices
[maxmllhd,indmax] = max(ymax(:,1));
maxtt = ymax(indmax,2); % maxtt is the index of the max row, form 1-Npoints,
                        % theta(indmax) needs to be changed
```

```
thetatemp = ones(Npoints,1)*theta;
thetaincrease = linspace(lb(indmax),ub(indmax),Npoints)';
thetatemp(:,indmax) = thetaincrease;  %thetaincrease records the
                                      %increasement for every parameter
Tthetatemp = transtheta(thetatemp,1,getdist);
TthetaP = Tthetatemp(maxtt,:);
```

摘自 7.17 节，Matlab 中天然气定价

主要的 Matlab 代码由 James Xu（2004）撰写，下面给出使用实时天然气数据的天然气期货定价.

estimateRData.m

```
clear; close all;
load gasTD_19920102_19991230.mat
ind=[];
for i = 1:length(tTDFS(:,2))
    if tTDFS(i,1)==tTDFS(i,2)
        ind=[ind;i];
    end
end
tTDFS(ind,2)=tTDFS(ind,2)+1;

type = 'A';
FWDtype = 'org';

Ntheta = 12;
NthetaX = 7;
period = 251;

[X1,ft1,RAtheta1,VsqdiffF,flag,output] = XRAthetafromtTDFS(tTDFS,NthetaX,
  Ntheta,period,FWDtype,type);
disp(RAtheta1)

thetaX0 = ones(1,NthetaX) * 0.1;
opt = optimset('maxfunevals',2e5,'tolx',1e-10,'maxiter',3000,'gradobj',
  'off','largescale','on','display','iter');

[thetaX1,Vasq1,flag1,output1] = fminsearch(@allsquares1,thetaX0,opt,
  X1,period,type);
[asq1,thetaXend1,sigmat1] = allsquares1(thetaX1,X1,period,type);
theta_est = [thetaXend1(1:NthetaX), RAtheta1(NthetaX+1:end)];
RAtheta_est = [RAtheta1(1:2),thetaXend1(3:NthetaX),
  RAtheta1(NthetaX+1:end)];

disp([theta_est;RAtheta_est])

%
% drawXLft;

L1 = theta_est(2);
figure(7);
hold off
plot(S)
hold on
plot(ft1+L1,'r')
%legend('X', 'estimated L');
title('S and f(t)+L');
```

```
hold off

tTDFSX = tTDFS;

dataname = strcat('allRData_',FWDtype,'_',type);
eval(['save ' dataname]);
```

all squares1. m

```
function [asq,thetaXend,sigmat] = allsquares1(thetaX,X,period,type)

NthetaX = length(thetaX);
t = 1:length(X);

c = thetaX(3);
sigmat = exp(c*ones(size(t)));
if NthetaX>3
    Npsgm = (NthetaX-3)/2; % number of periods in sigma(t);
    lambda = thetaX(4:3+Npsgm);
    omega = thetaX(4+Npsgm:NthetaX);
    for k = 1:Npsgm
        sgtmp = lambda(k)*cos((2*pi*k*t)/period)+
          omega(k)*sin((2*pi*k*t)/period);
        sigmat = sigmat.*exp(sgtmp);
    end
end

switch type
    case 'A'
        sgx = (sigmat(1:end-1).^2);
    case 'B'
        sgx = (sigmat(1:end-1).^2).*X(1:end-1);
    case 'C'
        sgx = (sigmat(1:end-1).^2).*(X(1:end-1).^2);
end

B = sum((X(1:end-1).^2)./sgx);
C = sum((-X(1:end-1))./sgx);
D = sum((X(2:end)-X(1:end-1)).*X(1:end-1)./sgx);
E = sum(1./sgx);
F = sum((X(1:end-1)-X(2:end))./sgx);

alpha = (D*E-C*F)/(C^2-B*E);
L = -B/C - D/(C*alpha);

Bt = (alpha*(L-X(1:end-1))+X(1:end-1)-X(2:end));
Dc = sum(1-(Bt.^2)./sgx);
Dlambda = 0;
Domega = 0;
if NthetaX>3
    Dlambda = zeros(1,Npsgm);
    Domega = zeros(1,Npsgm);
    for k = 1:Npsgm
        Dlambda(k) = sum((1-(Bt.^2)./sgx).*(cos(2*pi*k*t(1:end-1)/period)));
        Domega(k) = sum((1-(Bt.^2)./sgx).*(sin(2*pi*k*t(1:end-1)/period)));
    end
end

asq = Dc^2 + norm(Dlambda)^2 + norm(Domega)^2;
thetaXend = thetaX;
thetaXend(1) = alpha;
thetaXend(2) = L;
```

XRAthetafrontTDFS.m

```
function [X,ft,RAtheta,VsqdiffF,flag,output] = XRAthetafromtTDFS(tTDFS,
  NthetaX,Ntheta,period,FWDtype,type)
% compute L and RAtheta from tTDFS matrix-data-file.

noft = length(unique(tTDFS(:,1))); % length of t
nofFTi = size(tTDFS,1)/noft; % number of Forwards for 1 fixed t
tt = reshape(tTDFS(:,1),nofFTi,noft);
TT = reshape(tTDFS(:,2),nofFTi,noft);
TTD = reshape(tTDFS(:,3),nofFTi,noft);
FF = reshape(tTDFS(:,4),nofFTi,noft);
SS = reshape(tTDFS(:,5),nofFTi,noft);

iternum = 1000; % the number of iteration
opt = optimset('maxfunevals',2e5,'tolx',1e-10,'maxiter',iternum,'gradobj',
  'on','largescale','on','display','iter');
RAtheta0 = initRAtheta(SS,NthetaX,Ntheta);
% RAtheta0 = [0.0110   -0.0237   -0.0141    0.0498    0.0004    0.0006
%   0.8765 0.0002    0.1500    0.0600    0.0100   -0.0500   -0.0301
%  -0.0121];
[RAtheta,VsqdiffF,flag,output] = fminsearch(@sqdiffF,RAtheta0,opt,
  NthetaX,tt,TT,TTD,FF,SS,period,FWDtype,type);
Ntry = 4 ;% Number of restarting times
j = 0;
while flag <=0 & j<Ntry
    RAtheta = RAtheta*1.05;
    [RAtheta,VsqdiffF,flag,output] = fminsearch(@sqdiffF,RAtheta,opt,
      NthetaX,tt,TT,TTD,FF,SS,period,FWDtype,type);
    j = j+1;
end


[sqdF,X,ft] = sqdiffF(RAtheta,NthetaX,tt,TT,TTD,FF,SS,period,FWDtype,type);


function RAtheta0 = initRAtheta(SS,NthetaX,Ntheta);
RAtheta0 = [0.0100  zeros(1,NthetaX-1)  0.05*ones(1,Ntheta-NthetaX)]*1.5;
%  RAtheta0 = [ 0.0110    2.3000    -2.5000    0.4000    0.1000   -0.2000
%   -0.0700    -0.0001 0.1500    0.0600    0.0500    -0.0300]*1.5;
```

用下面的代码来初始化数据和初始的参数估计.

```
function RAtheta0 = initRAtheta(SS,NthetaX,Ntheta);
RAtheta0 = [0.0100 zeros(1,NthetaX-1) 0.05*ones(1,Ntheta-NthetaX)]*1.5;
%  RAtheta0 = [ 0.0110    2.3000    -2.5000    0.4000    0.1000   -0.2000
  -0.0700    -0.0001 0.1500    0.0600    0.0500    -0.0300]*1.5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% draw results
type = 'C';
FWDtype = 'org';
dataname = strcat('allRData_',FWDtype,'_',type);
eval(['load ' dataname]);

ft1 = ft1(1:length(X1));
sigmat1 = sigmat1(1:length(X1));

alpha = theta_est(1);
L = theta_est(2);
```

```
c = theta_est(3);
lambda1 = theta_est(4);
lambda2 = theta_est(5);
omega1 = theta_est(6);
omega2 = theta_est(7);
b = theta_est(8);
beta1 = theta_est(9);
beta2 = theta_est(10);
eta1 = theta_est(11);
eta2 = theta_est(12);

n = length(S);
X = S-ft1;
Xt = X(1:end-1);
Xtp = X(2:end);
switch type
  case 'A'
    sgx = sigmat1(1:end-1);

    case 'B'
        sgx = sigmat1(1:end-1).*sqrt(Xt);
    case 'C'
        sgx = sigmat1(1:end-1).*Xt;
end

M = 30; % rows of grids
N = 16; % columns of grids


fig = figure(10);
scz = get(0,'screensize');
figsize = [scz(1)+130,scz(2)+60,scz(3)-260,scz(4)-160];
set(fig,'position',figsize);

hold off;
dx = (alpha*(L-Xt)+Xt-Xtp)./sgx;

yy = 2; xx = 1;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
axis off;
text(1,1,'A)','fontsize',12);

yy = 1:6; xx = 3:8;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
[Nx,Px] = hist(dx,0.025*n);
hist(dx,0.025*n);
lim = max(abs(Px)*1.05);
xlim = [-lim, lim];
set(gca,'xlim',xlim)
h = findobj(gca,'Type','patch');
set(h,'FaceColor','w')
hold on;
a = linspace(-lim,lim,100);
b = 1/(sqrt(2*pi))*exp(-(a.^2)/(2));
c = b*n*(Px(2)-Px(1));
plot(a,c,'linewidth',2)
hold off;
```

```
yy = 2; xx = 9;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
axis off;
text(1,1,'B)','fontsize',12);

yy = 1:6; xx = 11:16;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
qqplot(dx);
xlim = [-3.7, 3.7];
ylim = [-9, 9];
set(gca,'xlim',xlim,'ylim',ylim);
title('')
xlabel('')
ylabel('')
yy = 12; xx = 1;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
axis off;
text(1,1,'C)','fontsize',12);

% switch type
%     case 'A'
%         rateb = [0.8, 0.15,   0.03, 0.3, 1.5, 0.6, 3.0,    0,0,0,0,0];
%     case 'B'
%         rateb = [0.8, 0.1,    0.02, 0.3, 1.5, 0.6, 3.0,    0,0,0,0,0];
%     case 'C'
%         rateb = [0.8, 0.2,    0.02, 0.3, 1.5, 0.6, 4.0,    0,0,0,0,0];
% end
% lb = theta_est - abs(theta_est).*rateb;
% ub = theta_est + abs(theta_est).*rateb;

switch type
    case 'A'
        stderr = [0.0046    0.0788    0.0151    0.0209    0.0241
0.0223    0.0210  0  0  0  0  0 ];
    case 'B'
        stderr = [0.0043    0.0858    0.0144    0.0242    0.0238
0.0242    0.0239  0  0  0  0  0];
    case 'C'
        stderr = [0.0043    0.0900    0.0127    0.0227    0.0210
0.0252    0.0201  0  0  0  0  0];
end
lb = theta_est - stderr;
ub = theta_est + stderr;

Npoints = 16;
drawslices(theta_est,NthetaX,lb,ub,t,S,ft1,period,Npoints,M,N,type)

Nsim = 250;
[simS,simX,simft,simsigmat,ts]=SimSX(S(1),theta_est,NthetaX,1,length(S),
  Nsim,period,type);

yy = 17; xx = 1;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
axis off;
text(1,1,'D)','fontsize',12);
```

```
yy = 16:19; xx = 3:16;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
[haxes,hline1,hline2] = plotyy(ts,ft1,ts,sigmat1);
axes(haxes(1))
ylabel('f(t)')
set(gca,'xlim',[1,length(S)+50])
axes(haxes(2))
ylabel('sigma(t)')
set(gca,'xlim',[1,length(S)+50])
set(hline2,'LineStyle','--')
%       legend('sigmat1',0);
%        legend('ft',0);
hold off;

yy = 24; xx = 1;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
axis off;
text(1,1,'E)','fontsize',12);

sortS = sort(simS);
simS_05p = sortS(ceil(size(simS,1)*0.05),:);
simS_95p = sortS(floor(size(simS,1)*0.95),:);
yy = 23:30; xx = 3:16;
subpp = subplotposition(M,N,yy,xx);
subplot(M,N,subpp);
plot(ts,S,'r','linewidth',1.5);
hold on;
plot(ts,simS(2,:),'b');
plot(ts,simS_05p,'g-.')
plot(ts,simS_95p,'m:');
xlabel('t');
ylabel('Price');
legend('real spot prices','1 path of simulation','5% percentile','95%
  percentile',2);
title('real spot prices vs. simulated spot price');
ylim = [0, max([simS(2,:),S])+0.3];
set(gca,'xlim',[1,length(S)+50],'ylim',ylim);
hold off;

% text(200,300,'hello')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [0.3 1 8.5 7.5]);
```

Minusloglkhood.M

```
function [mllhd,fis] = minusloglkhood(theta,NthetaX,t,S,ft,period,type)

Ntheta = length(theta);
X = S-ft;
t = t(:)';
t = t(1:end-1);
X = X(:)';

Xtp = X(2:end);
Xt = X(1:end-1);
alpha = theta(1);
L = theta(2);
```

```
c = theta(3);
sigmat = exp(c*ones(size(t)));
if NthetaX>3
    Npsgm = (NthetaX-3)/2; % number of periods in sigma(t);
    lambda = theta(4:3+Npsgm);
     omega = theta(4+Npsgm:NthetaX);
     for k = 1:Npsgm
         sgtmp = lambda(k)*cos((2*pi*k*t)/period)+
           omega(k)*sin((2*pi*k*t)/period);
         sigmat = sigmat.*exp(sgtmp);
     end
end

switch type
     case 'A'
             H1t = ones(size(Xt));
     case 'B'
             H1t = sqrt(Xt);
     case 'C'
             H1t = Xt;
end

     A1 = log(sqrt(2*pi)*sigmat.*H1t);
     A2 = ((alpha*(L-Xt)+Xt-Xtp).^2)./(2*(sigmat.^2).*(H1t.^2));

logfis = -A1-A2;

     mllhd = -sum(logfis);
```

forwardcurve.m

```
function [Ftis,ftiTis] = forwardcurve(RAtheta,NthetaX,Sti,ti,Tis,TDis,
  period,FWDtype,type)
% This function simulate forward price from given Spot price and L

Ntheta = length(RAtheta);

alpha1 = RAtheta(1);
L1 = RAtheta(2);

Tis = Tis(:)';
ft = zeros(1,max(max(Tis)));
if Ntheta > NthetaX % if the length of theta is larger than 4, we have
                    % included the seasonal factors
    t_of_ft = 1:max(max(Tis));
    b = RAtheta(NthetaX+1);
    ft = b*t_of_ft;
    if FWDtype == 'int'
        int_ft_TTTTD = 0.5*b*(Tis+TDis);
    end
    Npft = (Ntheta-NthetaX-1)/2;
    if Npft ~= 0
        beta = RAtheta(NthetaX+2:Ntheta-Npft);
        eta = RAtheta(Ntheta-Npft+1:end);
        for i = 1:Npft
            Pi = 2*pi*i/period;
            ftmp = beta(i)*cos(Pi*t_of_ft)+eta(i)*sin(Pi*t_of_ft);
            ft = ftmp + ft;
            if FWDtype == 'int'
            int_ft_TTTTD = int_ft_TTTTD + beta(i)*(sin(Pi*TDis)-
```

```
                   sin(Pi*Tis))./(Pi*(TDis-Tis))...
                                             + eta(i)*(cos(Pi*Tis)-
                                               cos(Pi*TDis))./(Pi*(TDis-Tis));
            end
        end
    end
end
ft = ft(:)';

switch type
    case {'A','B','C'}
        fti = ft(ti);
        fTis = zeros(size(Tis)); fTis(:) = ft(Tis); fTis = fTis(:)';
        if FWDtype == 'int'
            Aa = (exp(alpha1*(ti-Tis))-exp(alpha1*(ti-TDis)))./
              (alpha1*(TDis-Tis));
            Ftis = int_ft_TTTTD + L1+ (Sti-L1-fti).*Aa;
        else
            Ftis =  (L1+fTis) + (Sti-L1-fti).*exp(alpha1*(ti-Tis));
        end
end

ftiTis = [fti,fTis];

factor = 0;
Ftis = Ftis.*(1+factor*randn(size(Ftis))); % randomized Forward price,
                                           % controlled by 'factor'
```

sqdiffF.m

```
function [sqdF,X,ft] = sqdiffF(RAtheta,NthetaX,tt,TT,TTD,FF,SS,period,
  FWDtype,type)
% Compute the sum of squares of the differences between
% real Forward prices and those given by F(alpha1,mu1,t,T,S).
% We get the 'L1', 'alpha1' and 'mu1' via minimizing 'sqdF'.
% All the input arguments tt,TT,FF and SS are matrices, e.g., 12-by-500

Ntheta = length(RAtheta);

switch type
    case {'A','B','C'}

        alpha1 = RAtheta(1);
        L1 = RAtheta(2);
        ft = zeros(1,max(max(TT)));
        if Ntheta > NthetaX % if the length of theta is larger than that
                            % about X, we have included the seasonal
                            % factors
            t_of_ft = 1:max(max(TT));
            b = RAtheta(NthetaX+1);
            ft = b*t_of_ft;
            if FWDtype == 'int'
                int_ft_TTTTD = 0.5*b*(TT+TTD);

            end
            Npft = (Ntheta-NthetaX-1)/2;
            if Npft ~= 0
                beta = RAtheta(NthetaX+2:Ntheta-Npft);
                eta = RAtheta(Ntheta-Npft+1:end);
```

```
                for i = 1:Npft
                    Pi = 2*pi*i/period;
                    ftmp = beta(i)*cos(Pi*t_of_ft)+eta(i)*sin(Pi*t_of_ft);
                    ft = ftmp + ft;
                    if FWDtype == 'int'
                        int_ft_TTTTD = int_ft_TTTTD + beta(i)*(sin(Pi*TTD)-
                          sin(Pi*TT))./(Pi*(TTD-TT))...
                                       + eta(i)*(cos(Pi*TT)-
                                         cos(Pi*TTD))./(Pi*(TTD-TT));
                    end
                end
            end
        end
        fftt = zeros(size(tt)); fftt(:) = ft(tt);
        ffTT = zeros(size(tt)); ffTT(:) = ft(TT);
        if FWDtype == 'int'
            Aa = (exp(alpha1*(tt-TT))-exp(alpha1*(tt-TTD)))./
              (alpha1*(TTD-TT));
            FF2 = int_ft_TTTTD + L1+ (SS-L1-fftt) * Aa; % FF2 is the
                                                        % recomputed FF
                                                        % from given but
                                                        % unreal RAtheta
        else
            FF2 = (L1+ffTT) + (SS-L1-fftt).*exp(alpha1*(tt-TT );
        end

        sqdF = sum(sum((FF-FF2).^2));

        XX = SS-fftt;
        X = XX(1,:);
end
```

SimFWDmat. m

```
function tTDFSX = SimFWDmat(RAtheta_sim,NthetaX,S,X,tt,TT,TTD,period,
  FWDtype,type)
% This function simulate forward price from given Spot price
% almul_sim is the given risk-neutralized parameter containing alpha1 and
% mu1 and seasonal factors
% tt is start time, TT is the beginning of delivery time, TTD is the end
% of delivery time.

Ntheta = length(RAtheta_sim);

alpha1 = RAtheta_sim(1);
L1 = RAtheta_sim(2);

ft = zeros(1,max(max(TT)));
if Ntheta > NthetaX % if the length of theta is larger than 4, we have
                    % included the seasonal factors
     t_of_ft = 1:max(max(TT));
     b = RAtheta_sim(NthetaX+1);
     ft = b*t_of_ft;
     if FWDtype == 'int'
         int_ft_TTTTD = 0.5*b*(TT+TTD);
     end
     Npft = (Ntheta-NthetaX-1)/2;
     if Npft ~= 0
         beta = RAtheta_sim(NthetaX+2:Ntheta-Npft);
```

```
        eta = RAtheta_sim(Ntheta-Npft+1:end);
        for i = 1:Npft
            Pi = 2*pi*i/period;
            ftmp = beta(i)*cos(Pi*t_of_ft)+eta(i)*sin(Pi*t_of_ft);
            ft = ftmp + ft;
            if FWDtype == 'int'
                int_ft_TTTTD = int_ft_TTTTD + beta(i)*(sin(Pi*TTD)-
                  sin(Pi*TT))./(Pi*(TTD-TT))...
                                                  + eta(i)*(cos(Pi*TT)-
                                                    cos(Pi*TTD))./(Pi*(TTD-TT));
            end
        end
    end
end
ft = ft(:)';
tTDFSX(:,1) = tt(:);
tTDFSX(:,2) = TT(:);
tTDFSX(:,3) = TTD(:);

len = length(tt);
SS = zeros(len,1);
XX = zeros(len,1);

i = 1; j = 1;
SS(i) = S(j);
XX(i) = X(j);
for i = 2:len
    if tt(i)==tt(i-1)
        SS(i) = SS(i-1);
        XX(i) = XX(i-1);
    else
        j = j+1;
        SS(i) = S(j);
        XX(i) = X(j);
    end
end
tTDFSX(:,5) = SS;
tTDFSX(:,6) = XX;

switch type
    case {'A','B','C'}
        fftt = zeros(size(tt)); fftt(:) = ft(tt);
        ffTT = zeros(size(tt)); ffTT(:) = ft(TT);
        if FWDtype == 'int'
            Aa = (exp(alpha1*(tt-TT))-exp(alpha1*(tt-TTD)))./
              (alpha1*(TTD-TT));
            Fprice = int_ft_TTTTD + L1+ (SS-L1-fftt).*Aa;
        else
            Fprice =  (L1+ffTT) + (SS-L1-fftt).*exp(alpha1*(tt-TT));
        end

end

factor = 0;
tTDFSX(:,4) = Fprice.*(1+factor*(1-2*rand(size(Fprice)))); % randomized
                                                           % Forward price,
                                                           % controlled by
                                                           % 'factor'
```

allestimate.m

```
ntimes = 50;

type = 'C';
alltheta_est = zeros(ntimes,24);

for i = 1:ntimes

    SimAllData;
    estimate;
    alltheta_est(i,1:12) = theta_est;
    alltheta_est(i,13:24) = RAtheta_est;

end

save allestimate_C theta_sim RAtheta_sim alltheta_est
```

drawPrices

```
load gasTD_19920102_19991230.mat
S(end+1) = S(end);
SS = reshape(S,251,8);
SS = SS';
% plot(1:251,SS(1:8,:),'g--');
% hold on;
% subplot(1,2,1)
figure(3);
hold off;
plot(mean(SS(1:8,:)),'b','LineWidth',2)

title('natural gas spot average daily prices over 8 years')
axis([0,260,1.7,2.7]);
xlabel('time (day)')
ylabel('price (dollors per MMBTU)')
hold off;
%
%
% DATESTR(729895,2)
load novagasmonthly
dn = datenum(1997,12,30);
ind = find(mdata(:,1)==dn);
fp = mdata(ind,4)';
delivdate = mdata(ind,2)';
% subplot(1,2,2)
figure(4);
plot(delivdate,fp);
title('natural gas futures prices')
axis([dn-50,max(mdata(ind,2))+50,2,2.7]);
xlabel('delivery time (mm/dd/yy)')
ylabel('price (dollors per MMBTU)')

xtick = [dn, delivdate(11),delivdate(23),delivdate(35)];
xticklabel = {'01/01/98','01/01/99','01/01/00','01/01/01'};
set(gca,'xtick',xtick, 'XTickLabel',xticklabel)

% load gasTD_19920102_19991230.mat
% fig3 = figure(3)
% plot(S)
% title('AECO gas prices')
% xlabel('time  (days)')
% ylabel('price (dollors per MMBTU)')
% xlim = [0,2100];
% set(gca,'xlim',xlim);
```

estlfMR.m

```
function [alpha, L, sigma] = estlfMR(S,type)
switch type
    case 'A'
        A = ones(size(S));
        A = A(1:end-1);
    case 'B'
        A = sqrt(S(1:end-1));
    case 'C'
        A = S(1:end-1);
end
St = S(1:end-1);
Stp = S(2:end);

B = sum((St.^2)./(A.^2));
C = sum((-St)./(A.^2));
D = sum((St.*(Stp-St))./(A.^2));
E = -sum(C);
F = sum(-1./(A.^2));
G = sum((Stp-St)./(A.^2));

alpha = (D*F-C*G)/(C*E-B*F);
L = -B/C - D/(C*alpha);
sigma = sqrt(mean(((alpha*(St-L)+(Stp-St)).^2)./(A.^2)));

% x = S(1:end-1); y = S(2:end);
% p = polyfit(x,y,1);
% z = p(1)*x + p(2);
% alpha = 1-p(1);
% L = p(2)/alpha;
% switch type
%     case 'A'
%         sigma = std(y-z);
%     case 'B'
%         sigma = std((y-z)./sqrt(x));
% end
```

SimFWDtTD.m

```
function [tt,TT,TTD,ttTTTTD] = SimFWDtTD(t,n,m,Tdiff,DelivLen)
% n is the number of T for 1 t
% m is the number of trading days in 1 month, usually being 30 here
% Tdiff is the difference of 2 succesive T's
% DelivLen is the length of delivery period

t = t(:)';
tt = repmat(t,n,1);

nofT = n + ceil(length(t)/m) + 3; % nofT is the number of all T's. To make
                                  % sure there are enough number of T's,
                                  % we add 3 up to it.
T0 = t(1)+Tdiff;
Ts = T0:Tdiff:(T0+nofT*Tdiff);
Ts = Ts + cumsum(round(rand(size(Ts)))); % Tdiff sometimes is 30,
                                         % sometimes is 31.
Ts = Ts(:);
TT = zeros(size(tt));
for i = 1:size(tt,2)
    ind = find(Ts>tt(1,i));
    ind = ind(1:n);
    TT(:,i) = Ts(ind);
end
tt =tt(:);
TT =TT(:);
```

```
TTD = TT+DelivLen;
ttTTTTD = [tt,TT,TTD];
```

SimSX.m

```
function [S,X,ft,sigmat,t]=SimSX(S0,theta,NthetaX,t0,TEnd,N,period,type)
% to get 2 matrices containing S's, X's and L's simulation respectively.
% S0 is the initial value of S
% N is the number of simulation paths
% t0 is the beginning of the time, TEnd is end
% the time length of every simulation is TEnd-t0+1
% dt is the time-step
% theta = [alpha, L, c, lambda..., omega..., b, beta..., eta...]
dt = 1;
t = t0:TEnd;
Ntheta = length(theta);
alpha = theta(1);
L = theta(2);
c = theta(3);
sigmat = exp(c*ones(size(t)));
if NthetaX>3
    Npsgm = (NthetaX-3)/2; % number of periods in sigma(t);
    lambda = theta(4:3+Npsgm);
    omega = theta(4+Npsgm:NthetaX);
    for k = 1:Npsgm
        sgtmp = lambda(k)*cos((2*pi*k*t)/period)+omega(k)*
          sin((2*pi*k*t)/period);
        sigmat = sigmat.*exp(sgtmp);
    end
end
ft = 0;
if Ntheta>NthetaX % if the length of theta is larger than that about X,
                  % we have included the seasonal factors
    b = theta(NthetaX+1);
    ft = b*t;
    Npft = (Ntheta-NthetaX-1)/2;
    if Npft ~= 0
        beta = theta(NthetaX+2:Ntheta-Npft);
        eta = theta(Ntheta-Npft+1:end);
        for i = 1:Npft
            ftmp = beta(i)*cos((2*pi*i*t)/period)+eta(i)*
              sin((2*pi*i*t)/period);
            ft = ftmp +ft;
        end
    end
    ft = ft(:)';
end
M = TEnd-t0+1; % column of the S matrix
X = zeros(N,M);
X(:,1) = S0 - ft(1);
W1=randn(N,M);

switch type
    case 'A'
        for j=2:M
            X(:,j) = X(:,j-1) + alpha*dt*(L-X(:,j-1)) +
              sigmat(j-1)*(dt^0.5)*W1(:,j-1);
        end
    case 'B'
```

```
        for j=2:M
            X(:,j) = X(:,j-1) + alpha*dt*(L-X(:,j-1)) +
              sigmat(j-1)*(dt^0.5)*sqrt(X(:,j-1)).*W1(:,j-1);
        end
    case 'C'
        for j=2:M
            X(:,j) = X(:,j-1) + alpha*dt*(L-X(:,j-1)) +
              sigmat(j-1)*(dt^0.5)*X(:,j-1).*W1(:,j-1);
        end
end

S = X + repmat(ft,N,1);
```

forwardmatch.m

```
% ss = input('====Do you want to see how the expectation of the simulated
%   spot price match the forward curve?(y/n) ','s');
% if strcmp(ss,'y')
scz = get(0,'screensize');
figsize = [scz(1)+130,scz(2)+60,scz(3)-260,scz(4)-160];
% maxF = max(tTDFSXL(:,4));
% minF = min(tTDFSXL(:,4));
fig = figure(8);
set(fig,'position',figsize);

loop1 = 0;
t1 = 1;
while 1
    while 1
        if loop1~=1
            getdate = input(['====Please input a number within ',
              num2str(t(1)),'~',num2str(t(end)),': '],'s');
        else
            getdate = '0';
            t1 = t1+1;
            pause(0.3);
        end

        if isempty(getdate)
            t1=t1+1;
        elseif getdate=='0'
            loop1 = 1;
        else
            t1 = str2num(getdate);
        end

        [ti,Tis,TDis,Sti,Ftis] = DataFilter(tTDFSX,t1);
        if isempty(ti)
            if loop1==1, return; end
            disp('There is no forwards w.r.t. the number you input.
              Choose again, please.');
        else
            break;
        end
    end
    %     if isempty(getdate), break; end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% sim %%%%%%%%%%%%%%%%
    N = 500;
    [Stis,Xtis,ftis,sigmatis,tis]=SimSX(Sti,RAtheta_est,NthetaX,ti,
      Tis(end),N,period,type);
```

```
sortS = sort(Stis);
simS_05p = sortS(ceil(size(Stis,1)*0.05),:);
simS_95p = sortS(floor(size(Stis,1)*0.95),:);

tiTis = [ti,Tis];
StiFtis = [Sti,Ftis];

subplot(30,1,[1,6]);
hold off
[haxes,hline1,hline2] = plotyy(tis,ftis,tis,sigmatis);
%        set(gca,'xtick',tT,'xticklabel',tT);
axes(haxes(1))
ylabel('f(t) (dollars)')
xlabel('time t (days)');
set(gca,'xtick',tiTis,'xticklabel',tiTis-tiTis(1));
axes(haxes(2))
ylabel('sigma(t)')
set(gca,'xtick',tiTis,'xticklabel',tiTis-tiTis(1));
set(hline2,'LineStyle','--')
title('f(t) and sigma(t)')

subplot(30,1,22:30);
hold off
plot(tiTis,StiFtis,'bo');
hold on;
% plot(tis,Stis(2,:),'k');
% plot(tis,simS_05p,'g-.')
% plot(tis,simS_95p,'m:');
plot(tiTis,StiFtis,'b','linewidth',1.5);
plot(tis,mean(Stis),'r');
set(gca,'xtick',tiTis,'xticklabel',tiTis-tiTis(1));
xlabel('time t (days)');
ylabel('Price (dollars)');
%      legend('One path of spot price simulation','5% percentile',
%      '95% percentile','Forward curve','Mean of spot price
%      simulations',0);
title('Expectation of simulated spot prices vs. the real
  futures curve');
hold off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% formula %%%%%%%%%%%%%%%%%%%
[myFtis,ftiTis] = forwardcurve(RAtheta_est,NthetaX,Sti,ti,Tis,TDis,
  period,FWDtype,type);

subplot(30,1,10:18);
hold off

tiTis = [ti,Tis];
StiFtis = [Sti,Ftis];
StimyFtis = [Sti,myFtis];
plot(tiTis,StiFtis,'bo');
hold on;
plot(tiTis,StiFtis,'b','linewidth',1.5);
plot(tiTis,StimyFtis,'ro');
plot(tiTis,StimyFtis,'r');
%        ylim = [minF,maxF];
%        ylim = [ylim(1)-eps,ylim(end)+eps];
set(gca,'xtick',tiTis,'xticklabel',tiTis-tiTis(1));
xlabel('time t (days)');
ylabel('Price (dollars)');
```

```
    %      legend('One path of spot price simulation','5% percentile',
    %      '95% percentile','Forward curve','Mean of spot price
    %      simulations',0);
    title('Futures curve from formula vs. the real forward curve');
    hold off;

    set(gcf, 'PaperPositionMode', 'manual');
    set(gcf, 'PaperUnits', 'inches');
    set(gcf, 'PaperPosition', [0.2 1.5 7 7.3]);

end

% end
% Compute the cumulative Normal distribution in terms of the error function.
function y = N1(x);
  y = (1+erf(x/sqrt(2)))/2;

% Compute the inverse cumulative Normal distribution in terms of the
% error function
function y = N1inv(x);
  y = sqrt(2)*erfinv(2*x-1);
```

参考文献

第 1 章

Audley, D., Chin, R., and Ramamuthy, S. (2002), "Term Structure Modeling," in *Interest Rate, Term Structure, and Valuation Modeling*. Edited by Fabozzi, F., John Wiley & Sons, Inc.

Chicago Board of Trade (2001), "Hedging a Fixed-Income Portfolio with Swap Futures," CBOT Interest Rate Swap Complex Series. *http://www.cbot.com/cbot/pub/page1/1,3248,1060,00.html.*

Fabozzi, F. and Mann, S. (Editors) (2001), *The Handbook of Fixed-Income Securities*. McGraw-Hill: New York, NY.

Fabozzi, F. (2002), *Interest Rate, Term Structure, and Valuation Modeling*. John Wiley & Sons, Inc.

Hull, J. (2005), *Fundamentals of Futures and Options Markets*. Prentice Hall.

James, J. and Webber, N. (2000), *Interest Rate Modeling*. John Wiley & Sons: Chichester, UK.

Jarrow, R. and Turbull, S. (2000), *Derivative Securities, 2nd ed.* South-Western Publishers: Cincinnati, Ohio.

Johnson, S.R. (2002), "Futures on Debt Positions," Williams School of Business, Xavier University, working paper. *http://www.academ.xu.edu/johnson/.*

Johnson, S.R. (2002), "Hedging Debt Positions with Futures and Options," Williams School of Business, Xavier University, working paper.

Johnson, S.R. (2004), *Bond Evaluation, Selection, and Management*. Blackwell Publishing.

Piennar, R. and Choudry, M. (2002), "Fitting the Term Structure of Interest Rates Using the Cubic Spline Methodology," in *Interest Rate, Term Structure, and Valuation Modeling*. Wiley & Sons, Inc., 157–185.

Strumeyer, G. (2005), *Investing in Fixed Income Securities: Understanding the Bond Market*. Wiley & Sons: Hoboken, NJ.

Waggoner, D. (1997), "Spline Methods for Extracting Interest Rate Curves from Coupon Bond Prices," working paper 97–10, Federal Reserve Bank of Atlanta.

第 2 章

Bouyè, E., Durrleman, V., Nikeghbali, A., Riboulet, G., and Roncalli, T. (2000), "Copulas for Finance: A Reading Guide and Some Applications," Groupe de Reserche Operationnelle, Credit Lynonnais, Paris, working paper.

Drouet Mari, D. and Kotz, S. (2001), *Correlation and Dependence*. Imperial College Press, London.

Embrechts, P., Lindskog, F., & McNeil, A. (2001), "Modelling Dependence with Copulas and Application to Risk Management," ETH Zurich, Department of Mathematics, working paper.

Galiani, S. (2003), "Copula Functions and Their Applications in Pricing and Risk Managing Multiname Credit Derivative Products," Masters Thesis, King's College, London.

Hull, J. and White, A. (2005), "The Perfect Copula," University of Toronto, working paper.

Joe, H. (1997), *Multivariate Models and Dependence Concepts*. Chapman and Hall, London.

Joe, H. and Xu, J.J. (1996), "The Estimation Method of Inference Function for Margins for Multivariate Models," Dept. of Statistics, University of British Columbia, technical report, 1966.

Johnson, N.L. and Kotz, S. (1972), *Distribution in Statistics: Continuous Multivariate Distributions*. John Wiley & Sons, Inc.: New York.

Li, D. (1999), "On Default Correlation: A Copula Function Approach," The RiskMetrics Group, working paper, no. 99/07.

Lindskog, F., McNeil, A., and Schmock, U. (2001), "Kendall's Tau for Elliptical Distributions," ETH Zurich, Department of Mathematics, working paper.

Mashal, R. and Naldi, M. (2002), "Generalizing Asset Dependency: Application to Credit Derivatives Pricing," Quantitative Credit Research Quarterly, Lehman Brothers Inc., working paper.

Mashal, R. and Zeevi, A. (2002), "Beyond Correlation: Extreme Co-Movements Between Financial Assets," working paper, Columbia Graduate School of Business.

Meneguzzo, D. and Vecchiato, W. (2002), "Copula Sensitivity in Collateralized Debt Obligations and Basket Default Pricing and Risk Monitoring," Risk Management Dept., Intesa Bank, Mila, working paper.

Scaillet, O. (2000), "Nonparametric estimation of copulas for time series," IRES, working paper.

Schonbucher, P. and Schubert, D. (2001), "Copula-dependent default risk in intensity models," Department of Statistics, University of Bonn, working paper.

第 3 章

Archer, W. and Ling, D. (1995), "The Effect of Alternative Interest Rate Processes on the Value of Mortgage-Backed Securities," *Journal of Housing Research*, 6: 2, 285–314.

Bandic, I. (2002), "Pricing Mortgage-Backed Securities and Collateralized Mortgage Obligations," University of British Columbia, working paper.

Boudoukh, J., Richardson, M., Stanton, R., and Whitelaw, R. (1995), "A New Strategy for Dynamically Hedging Mortgage-Backed Securities," working paper.

Boudoukh, J., Richardson, M., Stanton, R., and Whitelaw, R. (1998), "The Pricing and Hedging of Mortgage-Backed Securities: A Multivariate Density Estimation Approach," working paper.

Boudoukh, J., Richardson, M., Stanton, R., and Whitelaw, R. (2003), "The Valuation and Hedging of Deferred Commission Asset-Backed Securities," working paper.

Chatterjee, S. (1999), "The ARM Prepayment Model," *Quantitative Perspectives*. Andrew Davidson & Co., February.

Chatterjee, S. (2005), "Fixed-Rate Home Equity Loan Prepayment Model," *Quantitative Perspectives*. Andrew Davidson & Co., September.

Fabozzi, F., Richard, S., and Horwitz, D. (2002), "Monte Carlo Simulation/OAS Approach to Valuing Residential Real Estate-Backed Securities," in *Interest Rate, Term Structure, and Valuation Modeling*. Edited by Fabozzi, F., John Wiley & Sons, 443–468.

Gauseel, N. and Tamine, J. (2004), "Valuation of Mortgage-Backed Securities: From Optimality to Reality," Societe Generale Asset Management, working paper.

Johnson, S.R. (2002), "Analysis of Mortgage-Backed Securities: Monte Carlo Simulation Cash Flow Analysis," Xavier University, Williams School of Business. *http://www.academ.xu.edu/johnson/*.

Johnson, S.R. (2002), "Mortgage-Backed Securities," Xavier University, Williams School of Business. *http://www.academ.xu.edu/johnson/*.

Kalotay, A., Yang, D., and Fabozzi, F. (1991), "An Option-Theoretic Prepayment Model for Mortgages and Mortgage-Backed Securities," Andrew Kalotay Associates, Inc., working paper.

Kalotay, A., Yang, D., and Fabozzi, F. (1991), "When to Refinance: An Option-Based Approach," Andrew Kalotay Associates, Inc., working paper.

Kariya, T. and Kobayashi, M. (2000), "Pricing Mortgage-Backed Securities: A Model Describing the Burnout Effect," *Asia-Pacific Financial Markets*, 7, 189–204.

Kariya, T., Ushiyama, F., and Pliska, S. (2002), "A 3-Factor Valuation Model for Mortgage-Backed Securities (MBS)," working paper, April.

McConnell, J., and Singh, M. (1994), "Rational Prepayments and the Valuation of Collateralized Mortgage Obligations." *Journal of Finance*, 49, 891–921.

Obazee, P. (2002), "Understanding the Building Blocks for OAS Models," in *Interest Rate, Term Structure, and Valuation Modeling*. Edited by Fabozzi, F., John Wiley & Sons, 338–339.

Pliska, S. (2005), "Mortgage Valuation and Optimal Refinancing," Department of Finance, University of Illinois at Chicago, working paper.

Roll, R. and Scott, R. (1989), "Modeling Prepayments on Fixed-Rate Mortgage-Backed Securities," *Journal of Portfolio Management*, Spring, 73–82.

Sing, T.F., Ong, S.E., Fan, G., and Sirmans, C.F. (2001), "Cash Flow Swaps in Asset-Backed Securitization Transactions," Department of Real Estate, National University of Singapore, working paper.

Spahr, R. and Sunderman, M. (1991), "The Effect of Prepayment Modeling in Pricing Mortgage-Backed Securities," *Journal of Housing Research*, 3: 2, 381–399.

Stern, H. (2000), "ARM Home Equity Loan Prepayment Loan," *Quantitative Perspectives*. Andrew Davidson & Co., working paper.

Surkov, V. (2004), "Valuation of Mortgage-Backed Securities in a Distributed Environment," University of Toronto, Masters Thesis.

Titman, S., and Torous, W. (1989), Valuing Commercial Mortgages: An Empirical Investigation of the Contingent Claims Approach to Valuing Risky Debt. *Journal of Finance*, 44, 354–373.

第 4 章

Bluhm, C., Overbeck, L., and Wagner, C. (2003), *An Introduction to Credit Risk Modeling*. Chapman & Hall/CRC.

Boscher, H. and Ward, I. (2002), "Long or short in CDOs," *RISK Magazine*, June, 135–129.

Credit Suisse First Boston (1997), *CreditRisk+—A Credit Risk Framework*.

Duffie, D. and Garleanu, N. (2001), "Risk and Valuation of Collateralized Debt Obligations." *Financial Analysts Journal* (January/February), 41–59.

Galiani, S. (2003), "Copula Functions and Their Applications in Pricing and Risk Managing Multiname Credit Derivative Products," Masters Thesis, King's College, London.

Gibson, M. (2004), "Understanding the Risk of Synthetic CDOs." Trading Risk Analysis Section, Division of Research and Statistics, Federal Reserve Board, working paper.

Goodman, L. (2002), "Synthetic CDOs: An Introduction." *Journal of Derivatives*, (Spring), 60–72.

Gregory, J. and Laurent, J. (2002), "Basket Default Swaps, CDOs, and Factor Copulas," working paper.

Gupton, M., Finger, C., and Bhatia, M. (1997), CreditMetrics—Technical Document. Risk Management Research, Morgan Guaranty Trust Company.

Hull, J., and White, A. (2004), "Valuation of a CDO and an nth to Default CDS without Monte Carlo Simulation," Rotman School of Management, University of Toronto, working paper.

Isla, L. (2003), "European CDOs: Review and Outlook," *Structured Credit Research*, Lehman Brothers, April.

Lehnert, N., Altrock, F., Rachev, S., Truck, S., and Wilch, A. (2005), "Implied Correlation in CDO Tranches," preprint.

Li, D. (2000), "On default correlation: a copula function approach," *Journal of Fixed Income* (March), 115–118.

Li, D. and Liang, M. (2005), CDO^2 Pricing Using Gaussian Mixture Model with Transformation of Loss Distribution, Barclays Capital, Quantitative Analytics, Global Credit Derivatives, working paper.

McGinty, L, and Ahuluwalia, R. (2004), A Relative Value Framework for Credit Correlation. Credit Derivatives Strategy, JP Morgan.

Merton, R. (1974), "On the Pricing of Corporate Debt: The Risk Structure of Interest Rates." *Journal of Finance*, 29, 449–470.

Picone, D. (2003) "Collateralized Debt Obligations." City University Business School, London and Royal Bank of Scotland, working paper.

Schmidt, W. and Ward, I. (2002), "Pricing Default Basket," *RISK Magazine*, January, 111–114.

Schonbucher, P. (2003), *Credit Derivatives Pricing Models: Models, Pricing, and Implementation.* John Wiley & Sons: Chichester, UK.

Tavakoli, J. (2003), *Collateralized Debt Obligations and Structured Finance: New Developments in Cash and Synthetic Securitization*. John Wiley & Sons, Hoboken, NJ.

Vasicek, O. (1987), "Probability of loss on loan portfolio." Moody's KMV (can be downloaded at *http://www.moodyskmv.com/research/whitepaper/Probability_of_Loss_on_Loan_Portfolio.pdf*), working paper.

Willemann, S. (2004), "An evaluation of the base correlation framework for synthetic cdos," working paper.

第 5 章

Bluhm, C., Overbeck, L., and Wagner, C. (2003), *An Introduction to Credit Risk Modeling*. Chapman and Hall/CRC Financial Mathematics Series, Boca Raton.

Chen, R. and Sopranzetti, B. (2002), "The Valuation of Default-Triggered Credit Derivatives," Rutgers Business School, working paper.

Das, S., Freed, L., Geng, G., and Kapadia, N. (2002), "Correlated Default Risk," working paper.

Duffie, D. and Singleton, K. (1999), "Modeling Term Structures of Defaultable Bonds," *Review of Financial Studies*, 12(4), 687–720.

Galiani, S. (2003), "Copula Functions and Their Applications in Pricing and Risk Managing Multiname Credit Derivative Products," Masters Thesis, King's College, London.

Gregory, J. and Laurent, J. (2002), "Basket Default Swaps, CDOs, and Factor Copulas," working paper.

Houweling, P. and Vorst, T. (2002), "An Empirical Comparison of Default Swap Pricing Models," Erasmus University Rotterdam, working paper.

Hull, J, Predescu, M., and White, A. (2003), "The Relationship Between Credit Default Swap Spreads, Bond Yields, and Credit Rating Announcements," Rotman School of Management, University of Toronto, working paper.

Lando, D. (1998). "On Cox processes and credit risk securities," Department of Operations Research, University of Copenhagen, working paper.

Li, D. (1998), "Constructing a Credit Curve," in *Credit Risk, a RISK Special Report*, November, 40–44.

Mashal, R. and Naldi, M. (2002), "Pricing Multiname Credit Derivatives: A Heavy-Tailed Approach," *Quantitative Credit Research Quarterly*, Lehman Brothers, working paper.

Merton, R. (1973), "Theory of Rational Option Pricing." *Bell Journal of Economics and Management Science* 4: 141–183.

Naldi, M. (2001), "Basket Default Swaps and the Credit Cycle," *Quantitative Credit Research Quarterly*, Lehman Brothers, June.

O'Kane. D. (2001), "Credit Derivatives Explained: Market, Products, and Regulations," *Structure Credit Research*, Lehman Brothers, March.

O'Kane, D. and Turnbull, S. (2003), "Valuation of Credit Default Swaps," *Fixed Income Quantitative Credit Research*, Lehman Brothers, April.

Romano, C. (2002), "Calibrating and Simulating Copula Functions: An Application to the Italian Stock Market," CIDEM, working paper.

Schmidt, W. and Ward, I. (2002), "Pricing Default Basket," *RISK Magazine*, January, 111–114.

Schonbucher, P.J. (2003), *Credit Derivatives Pricing Models: Models, Pricing, and Implementation.* John Wiley & Sons: Chichester, UK.

Zheng, C.K. (1999), "Default Implied Volatility for Credit Spread," Morgan Stanley, working paper.

第 6 章

Alaton, P., Djehiche, B., and Sillberger, D (2000), "On Modeling and Pricing Weather Derivatives," working paper.

Basawa, I. And P. Rao (1980), *Statistical Inference for Stochastic Processes*, Academic Press, New York.

Benth, F. and Šaltytė-Benth, J. (2005), "The Volatility of Temperature and Pricing of Weather Derivatives," University of Oslo, March.

Campbell, S. and Diebold, F. (2002), "Weather Forecasting for Weather Derivatives," reprint, December.

Cao, M., Li, A., and Wei, J. (2003), "Weather Derivatives: A New Class of Financial Instruments," University of Toronto, Rotman School of Management.

Dornier, F. and Queruel, M. (2000). "Caution to the Wind." *Risk, Energy, and Power Risk Management*, August.

Engle, R. (1982). "Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of the United Kingdom Inflation." *Econometrica*, 50.

Jewson, S. and Caballero, R. (2003), "The Use of Weather Forecasts in the Pricing of Weather Derivatives," Meteorological Applications, 1–13.

Jovin, E. (1998), "Advances on the Weather Front (An Overview of the U.S. Weather Derivatives Market, Global Energy Risk)," 212(9), pg. S4 (2).

Kaminski, V. (1998), "Pricing Weather Derivatives (Global Energy Risk)." *Electrical World*, 212(9), pg. S4 (2)

Platen, E. and West, J. (2004), "Fair Pricing of Weather Derivatives." University of Technology Sydney, 1–27, July.

Roustant, O., Laurent, J., Bay, X., and Carraro, L. (2003), "A Bootstrap Approach to the Price Uncertainty of Weather Derivatives," working paper.

Zeng, L. (2000), "Pricing Weather Derivatives." *Journal of Risk Finance*, 72–78, Spring.

第 7 章

Barlow, M., Gusev, Y., and Lai, M. "Parameter Estimation of Multifactor Models in Power Markets," preprint.

Barone-Adesi, G. and Gigli, A. (2002), "Electricity Derivatives." Institute of Finance-USI, Italy, unpublished working paper, 1–15.

Carmona, R. and Touzi, N. (2003), "Optimal Multiple Stopping and Valuation of Swing Options," working paper, May.

Chacko, G. and Viceria, G. (2001). "Spectral GMM Estimation of Continuous-time Processes," Graduate School of Business Administration, Harvard University, Tech. Report.

Clement, D. and P. Protter (2002), "An Analysis of a Least Squares Regression Method for American Option Pricing," *Finance and Stochastics*, 6, 449–471.

Clewlow, L. and Strickland, C. (2000), *Energy Derivatives: Pricing and Risk Management*. Lacima Publications.

Deng, S. (1999), "Stochastic Models of Energy Commodity Prices and Their Applications: Mean-Reversion with Jumps and Spikes," Georgia Institute of Technology, working paper.

Doerr, U. (2003), "Valuation of Swing Options and Exercise and Examination of Exercise Strategies by Monte Carlo Techniques," Masters Thesis, Christ Church College. University of Oxford, September.

Duffie, D., Pan, J., and Singleton, K. (2000), "Transform Analysis and Asset Pricing for Affine Jump-Diffusions." *Econometrica*, 68, 1343–1376.

Energy Information Administration (2002), "Derivatives and Risk Management in the Petroleum, Natural Gas, and Electricity Industries," U.S. Department of Energy, Washington D. C., October, 1–87.

Eydeland, A. and Geman, H. (1999), "Fundamentals of Electricity Derivatives, in: Energy Modelling & the Management of Uncertainty," *Risk Publications*, London, 35–43.

Gibson, R. and Schwartz, E. (1990), "Stochastic Convenience Yield and the Pricing of Oil Contingent Claims." *Journal of Finance*, 45: 3, July, 959–976.

Goto, M. and Karolyi, G. A. (2003), "Understanding Electricity Price Volatility Within and Across Markets," The Central Research Institute of Electric Power Industry in Japan and the Dice Center for Financial Economics, Tech. Report, 2003.

Jaillet, P., Ronn, E., and Tompaidis, S. (2004), "Valuation of Commodity-Based Swing Options," *Management Science*, Vol. 50, No. 7, 909–921.

Kaminski, V. (1997), "The Challenge of Pricing and Risk Managing Electricity Derivatives, in: The US Power Market," *Risk Publications*, London, 149–171.

Keppo, J. (2004), "Pricing of Electricity Swing Options," *Journal of Derivatives*, Vol. 11, 26–43.

Lari-Lavassani, A., Sadeghi, A., and Ware, T. (2001), "Modeling and Implementing Mean Reverting Price Processes in Energy Markets." Electronic Publications of the International Energy Credit Association.

Longstaff, F. and Schwartz, E. (2001), "Valuing American Options by Simulation: A Simple Least-Squares Approach," *Review of Financial Studies*, 14, 113–147.

Lucia, J. and Schwartz, E. (2001), "Electricity Prices and Power Derivatives: Evidence from the Nordic Power Exchange." University of Valencia, working paper.

Matyas, L. (1999). *Generalized Method of Moments Estimation*, Cambridge.

Meyer, J. (2004), "Valuation of Energy Derivatives with Monte Carlo," Masters Thesis, Christ Church College. University of Oxford, September.

Pilipovic, D. (1998), "Energy Risk: Valuing and Managing Energy Derivatives," preprint.

Pindyk, R. (1999), "The Long-Run Evolution of Energy Prices," Massachusetts Institute of Technology, unpublished working paper.

Pindyk, R. (2003), "Volatility in Natural Gas and Oil Markets," Massachusetts Institute of Technology, working paper, June.

Schwartz, E. (1997), "The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging." *Journal of Finance*, 53 (3), July, 923–973.

Stoft, S., Belden, T., Goldman, C., and Pickle, S. (1998), "Primer on Electricity Futures and Other Derivatives." Environmental Energy Technologies Division, Berkeley National Laboratory, University of California-Berkeley.

Villaplana, P. (2002), "Pricing Power Derivatives: A Two-Factor Approach Jump-Diffusion Approach." University of Pompeu Fabra, Barcelona, Spain, working paper.

Xiong, L. (2004), "Stochastic Models for Electricity Prices in Alberta," Masters Thesis, Department of Mathematics and Statistics, University of Calgary, September.

Xu, Z. (2004), "Stochastic Models for Gas Prices," Masters Thesis, Department of Mathematics and Statistics, University of Calgary, August.

第 8 章

Barone-Adesi, G. and Gigli, A. (2002), "Electricity Derivatives," working paper, Universita della Svizzera Italiana.

Bessimbinder, H. and Lemmon, M. (2002), "Equilibrium Pricing and Optimal Hedging in Electricity Forward Markets," *Journal of Finance*, 57.

Duffy, D. (2006), *Finite Difference Methods in Financial Engineering: A Partial Differential Equations Approach*. John Wiley & Sons: New York.

Eydeland, A. and Wolyniec, K. (2002), *Energy and Power Risk Management: New Developments in Modeling, Pricing, and Hedging*. John Wiley & Sons: New York.

Geman, H. (2005), *Commodities and Commodity Derivatives: Modeling and Pricing for Agriculturals, Metals, and Energy*. Wiley Finance: West Sussex.

Geman, H. and Roncoroni, A. (2006), "Understanding the Fine Structure of Electricity Prices," *Journal of Business*, 79.

Harrison, M.J. and Taksar, M. (1983), "Instantaneous Control of Brownian Motion," *Mathematics of Operations Research*.

Pirrong, C. (2006a), "Incorporating Outages into the Pirrong-Jermakyan Framework," working paper, University of Houston.

Pirrong, C. (2006b), "The Valuation of Power Options in a Pirrong-Jermakyan Framework," working paper, University of Houston.

Pirrong, C. and Jermakyan, M. (2006), "The Price of Power: The Valuation of Power and Weather Derivatives," working paper, University of Houston.

Pirrong, C. (2006), "Incorporating Outages in a Pirrong-Jermakyan Model," working paper, University of Houston.

Tikhinov, A. and Arsenin, V. (1977), *Solution of Ill-Posed Problems*. Halsted Press: New York.

第 9 章

Duffie, D. and M. Huang (1996), "Swap Rates and Credit Quality," *Journal of Finance*, 51(3), July, 921–949.

Duffie, D. and Singleton, K. (1997), "An Econometric Model of the Term Structure of Interest Rate Swap Yields," *Journal of Finance*, 52(3): 1287–1321.

Fan, G., Sing, T., Ong, S.E., and Sirmans, C. (2004), "Governance and Optimal Financing for Asset-Backed Securitization," *Journal of Property Investment & Finance*, Vol. 22, No. 5, 414–434.

Ong, S. and Sing, T. (2000), "Asset Securitization in Singapore: A Tale of Three Vehicles," *Real Estate Finance*, 17(2), 47–56.

Sing, T., Ong, S., and Sirmans, C. (2003), "Asset-Backed Securitization in Singapore: Value of Embedded Buy-Back Options," *Journal of Real Estate Finance & Economics*, Vol. 27, No. 2, 173–180.

Sing, T.F., Ong, S.E., Fan G.Z., and Sirmans, C.F. (2004), "Analysis of Credit Risks in Asset-Backed Securitization Transactions in Singapore." Special issue for Maastricht-Cambridge Symposium 2002, *Journal of Real Estate Finance & Economics*, Vol. 28, No. 2/3, 235–254.

附录 A

Black, F. and Karasinski, P. (1991), "Bond and Option Pricing When Short Rates Are Log-normal." *Financial Analysts Journal*, July: 52–59.

Heath, D., Jarrow, R., and Morton, A. (1992), "Bond Pricing and the Term Structure of Interest Rates: A New Methodology for Contingent Claims Valuation." *Econometrica*, 60(1): 77–105.

Hull, J. and White, A. (1994), "Numerical Procedures for Implementing Term Structure Models I: Single-Factor Models." *Journal of Derivatives*, 1: 21–32.

Jamshidian, F. (1991). "Bond and Option Evaluation in the Gaussian Interest Rate Model." *Research in Finance* 9: 131–170.

华章数学译丛

书 名	书号（ISBN）	作 者	译 者	出版日期	页数	价格
随机过程导论（原书第2版）	7-111-31544-5	（美）Gregory F. Lawler	张景肖	2010-09-30	178	36
线性代数（原书第8版）	7-111-31344-1	（美）Steven J.Leon	张文博 张丽静	2010-09-20	457	65
拓扑学基础及应用	7-111-28809-1	（美）Colin Adams	沈以淡	2010-04-09	330	59
数学建模（原书第4版）	7-111-27729-3	Frank R. Giordano; William P.Fox; Steven B.Horton; Maurice D.Weir	叶其孝 姜启源	2009-08-07	425	69
初等数论及其应用（原书第5版）	7-111-26520-7	Kenneth H. Rosen	夏鸿刚	2009-06-16	469	68
数学建模方法与分析（原书第3版）	7-111-26640-2	Mark M. Meerschaert	刘来福 杨淳 黄海洋	2009-05-21	261	39
托马斯大学微积分	7-111-25134-7	Joel Hass;Maurice D.Weir; George B. Thomas，Jr	李伯民	2009-03-02	862	118
代数	7-111-25356-3	Michael Artin	郭晋云	2008-12-04	472	69
实分析与概率论（原书第2版）	7-111-23480-7	R.M.Dudley	赵选民 孙浩	2008-06-19	374	55
数论概论（原书第3版）	7-111-23911-6	Joseph H. Silverman	孙智伟 吴克俭 卢青林 曹惠琴	2008-05-27	289	42
高等微积分(原书第2版)	7-111-22790-8	Patrick M. Fitzpatrick	金嘉华 顾长康	2007-12-28	432	55
抽象代数基础教程 (原书第3版)	7-111-21262-1	Joseph J. Rotman	李样明 冯明军	2007-11-23	456	65
小波与小波变换导论	7-111-21544-8	C. Sidney Burrus; Ramesh A. Gopinath; Haitao Guo	程正兴	2007-11-13	239	32
应用组合数学(原书第2版)	7-111-20934-8	Fred S.Roberts ; Barry Tesman	冯速	2007-05-24	570	69
概率与计算	7-111-20805-1	Michael Mitzenmacher;Eli Upfal	史道济 等	2007-04-19	294	39
组合数学教程（原书第2版）	7-111-20595-1	J. H. van Lint ; R. M. Wilson	刘振宏 赵振江	2007-04-02	369	49
线性代数（原书第7版）	7-111-20845-7	Steven J. Leon	张文博 张丽静	2007-02-14	434	58
实用偏微分方程(原书第4版)	7-111-20022-5	Richard Haberman	郇中丹 李援南 刘歆 宋燕红	2007-02-08	538	66
复分析基础及工程应用（原书第3版）	7-111-20020-9	E. B. Saff , A. D. Snider	高宗升 等	2007-01-17	388	55
高等近世代数	7-111-19160-9	Joseph J.Rotman	章亮	2006-12-08	754	89
偏微分方程教程(原书第2版)	7-111-19746-1	Nakhlé H. Asmar	陈祖墀 宣本金	2006-11-03	698	85
微分几何及其应用（原书第2版）	7-111-19269-9	John Oprea	陈智奇 李君	2006-09-20	352	49
微积分及其应用 (原书第8版)	7-111-18992-2	Marvin L.Bittinger	杨奇 毛云英	2006-07-10	582	69
金融时间序列分析	7-111-18386-X	Ruey S.Tsay	潘家柱	2006-04-29	355	39
概率论基础教程(原书第6版)	7-111-18378-9	（美）Sheldon Ross	赵选民 等	2006-04-19	362	42
拓扑学(原书第2版)	7-111-17507-7	James R.Munkres	熊金城 吕杰 谭枫	2006-03-21	405	58
数学分析（原书第2版）	7-111-18014-3	Tom M.Apostol	邢富冲 邢辰 李松洁 贾婉丽	2006-03-06	400	55
图论导引（原书第2版）	7-111-17780-0	Douglas B.West	李建中 骆吉洲	2006-03-01	474	65
时间序列分析的小波方法	7-111-17806-8	Donald B.Percival,Andrew T.Walden	程正兴 等	2006-02-23	562	69
实分析(原书第3版)	7-111-17703-7	H.L.Royden	叶培新	2006-01-16	290	42
微分方程与边界值问题（原书第5版）	7-111-16874-7	Dennis G.Zill, Michael R.Cullen	陈启宏 张凡 郭凯旋	2005-11-17	580	68
线性规划导论	7-111-17329-5	Leonid Nison Vaserstein, Christopher Cattelier Byrne	谢金星 姜启源 张立平 等	2005-11-17	253	33
实分析与复分析（原书第3版）	7-111-17103-9	（美）Walter Rudin	戴牧民 张更容 郑顶伟 李世余	2005-11-05	335	42
数值分析(原书第3版)	7-111-16845-3	David Kincaid, Ward Cheney	王国荣 俞耀明 徐兆亮	2005-09-29	639	75
线性代数及其应用(原书第3版)	7-111-16709-0	David C.Lay	刘深泉 洪毅 马东魁 郭国雄 刘勇平	2005-08-18	496	59
复分析(原书第3版)	7-111-16793-7	Lars V. Ahlfors	赵志勇 薛运华 杨旭	2005-07-29	257	33
数学建模方法与分析（原书第2版）	7-111-16440-7	（新西兰）Mark M. Meerschaert	刘来福 杨淳 黄海洋	2005-06-29	256	33
复变函数及应用（原书第7版）	7-111-15830-X	（美）James Ward Brown, Ruel V.Churchill	邓冠铁 等	2005-02-25	357	32
泛函分析（原书第2版）	7-111-14405-8	（美）Walter Rudin	刘培德	2004-09-07	340	35
金融数学（中文版）	7-111-13816-3	（美）Joseph Stampfli, Victor Goodman	蔡明超	2004-03-04	227	26
数学分析原理（原书第3版）	7-111-13417-6	（美）Walter Rudin	赵慈庚 蒋铎	2004-02-10	306	28

教师服务登记表

尊敬的老师：

您好！感谢您购买我们出版的＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿教材。

机械工业出版社华章公司为了进一步加强与高校教师的联系与沟通，更好地为高校教师服务，特制此表，请您填妥后发回给我们，我们将定期向您寄送华章公司最新的图书出版信息！感谢合作！

个人资料（请用正楷完整填写）

教师姓名		□先生 □女士	出生年月		职务		职称：□教授 □副教授 □讲师 □助教 □其他
学校		学院		系别			
联系电话	办公： 宅电： 移动：	联系地址及邮编					
		E-mail					
学历		毕业院校		国外进修及讲学经历			
研究领域							

主讲课程	现用教材名	作者及出版社	共同授课教师	教材满意度
课程： □专 □本 □研 人数： 学期：□春□秋				□满意 □一般 □不满意 □希望更换
课程： □专 □本 □研 人数： 学期：□春□秋				□满意 □一般 □不满意 □希望更换

样书申请			
已出版著作		已出版译作	
是否愿意从事翻译/著作工作 □是 □否		方向	
意见和建议			

填妥后请选择以下任何一种方式将此表返回：（如方便请赐名片）

地 址：北京市西城区百万庄南街1号 华章公司营销中心 邮编：100037

电 话：(010) 68353079 88378995 传真：(010)68995260

E-mail:hzedu@hzbook.com markerting@hzbook.com 图书详情可登录http://www.hzbook.com网站查询

金融衍生品建模

基于Matlab、C++和Excel工具

本书汇集了多位行业骨干和专家的各类开发成果与模型，是第一本通过三个开发平台——Matlab、C++和Excel建模复杂衍生品的书。书中详细介绍了重要的衍生品定价模型，讨论了如何建立有效的模型，并提供了一些有用的技巧和方法。本书着重强调怎样利用C++、Matlab和Excel对价格、贸易和对冲交易这些复杂的模型进行编码，旨在教会读者正确地开发和执行衍生品程序。

本 书 特 点

- 涵盖了所有重要的衍生品定价模型，包括信用衍生品、债务抵押债券（CDO）、资产支持证券（ABS）、固定收益证券，以及天气、电力、能源衍生品等。
- 列举了大量的Matlab、C++和Excel应用实例，以帮助读者在实践中正确运用这些模型。
- 提供了Matlab和C++的示例代码，这些代码都可以更改和扩展，以满足读者实际需要。

作 者 简 介

Justin London 拥有密歇根大学经济学和数学的学士学位、应用经济学的文科硕士学位，以及金融工程、计算机科学和数学的理科硕士学位。他是全球在线交易和金融技术公司的创始人，曾为贸易公司和他自己的定量咨询公司开发固定收益和股权模型。他曾在芝加哥一家大银行使用信用衍生品分析和管理银行企业贷款组合，而且还指导几家银行完成了自己的衍生品交易系统。

Modeling Derivatives Applications in Matlab, C++, and Excel

PEARSON

www.pearsonhighered.com

客服热线：(010) 88378991，88361066
购书热线：(010) 68326294，88379649，68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com



上架指导：数学



ISBN 978-7-111-31296-3

9 787111 312963

定价：65.00 元

封面设计·杨宇梅